Appendix

The following is the hard copy of the HDL code used in the V1495 for this thesis and is current as of the time of publication. This version of the HDL is intended for interfacing with only one VFAT breakout board (six VFATs) and the secondary segments of the Robinson Nugent Multipin connectors are for debugging purposes. The default clock is set to a PLL (Phase Lock Loop) of 31 MHz (set system frequency at JLab).

v1495usr.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY v1495usr IS
PORT(
A  : IN  std_logic_vector (31 DOWNTO 0);
B  : IN  std_logic_vector (31 DOWNTO 0);
GIN: IN  std_logic_vector (1 DOWNTO 0);
IDD: IN  std_logic_vector (2 DOWNTO 0);
IDE: IN  std_logic_vector (2 DOWNTO 0);
IDF: IN  std_logic_vector (2 DOWNTO 0);
LCLK  : IN  std_logic;
PULSE : IN  std_logic_vector (3 DOWNTO 0);
WnR: IN  std_logic;
nADS  : IN  std_logic;
nBLAST: IN  std_logic;
nLRESET  : IN  std_logic;
C  : OUT std_logic_vector (31 DOWNTO 0);
DIRDDLY  : OUT std_logic;
GOUT  : OUT std_logic_vector (1 DOWNTO 0);
SELD  : OUT std_logic;
SELE  : OUT std_logic;
```

```
SELF  : OUT std_logic;
SELG  : OUT std_logic;
START : OUT std_logic_vector (1 DOWNTO 0);
WR_DLY0  : OUT std_logic;
WR_DLY1  : OUT std_logic;
nINT  : OUT std_logic;
nLEDG : OUT std_logic;
nLEDR : OUT std_logic;
nOED  : OUT std_logic;
nOEDDLY0 : OUT std_logic;
nOEDDLY1 : OUT std_logic;
nOEE  : OUT std_logic;
nOEF  : OUT std_logic;
nOEG  : OUT std_logic;
nREADY: OUT std_logic;
nSTART: OUT std_logic_vector (3 DOWNTO 2);
D  : IN  std_logic_vector (31 DOWNTO 0);
DDLY  : INOUT  std_logic_vector (7 DOWNTO 0);
E  : OUT std_logic_vector (31 DOWNTO 0);
F  : IN  std_logic_vector (31 DOWNTO 0);
FPGA  : INOUT  std_logic_vector (3 DOWNTO 0);
LAD: INOUT  std_logic_vector (15 DOWNTO 0);
SPARE : INOUT  std_logic_vector (11 DOWNTO 0)
);

-- Declarations

END v1495usr;
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_misc.all;
USE ieee.std_logic_unsigned.all;


ARCHITECTURE struct OF v1495usr IS

-- Architecture declarations

-- Internal signal declarations
SIGNAL DDLY_IN  : std_logic_vector(7 DOWNTO 0);
SIGNAL DDLY_OE  : std_logic;
SIGNAL DDLY_OUT : std_logic_vector(7 DOWNTO 0);
```

```
SIGNAL DLO0_GATE: std_logic;
SIGNAL DLO0_OUT : std_logic;
SIGNAL DLO1_GATE: std_logic;
SIGNAL DLO1_OUT : std_logic;
SIGNAL FPGA_DIR : std_logic_vector(3 DOWNTO 0);
SIGNAL FPGA_IN  : std_logic_vector(3 DOWNTO 0);
SIGNAL FPGA_OUT : std_logic_vector(3 DOWNTO 0);
SIGNAL GREEN_PULSE : std_logic;
SIGNAL LAD_IN: std_logic_vector(15 DOWNTO 0);
SIGNAL LAD_OE: std_logic;
SIGNAL LAD_OUT  : std_logic_vector(15 DOWNTO 0);
SIGNAL PDL0_IN  : std_logic;
SIGNAL PDL0_OUT : std_logic;
SIGNAL PDL1_IN  : std_logic;
SIGNAL PDL1_OUT : std_logic;
SIGNAL PDL_DIR  : std_logic;
SIGNAL PDL_READ : std_logic_vector( 7 DOWNTO 0);
SIGNAL PDL_SEL  : std_logic;
SIGNAL PDL_WR: std_logic;
SIGNAL PDL_WRITE: std_logic_vector( 7 DOWNTO 0);
SIGNAL RED_PULSE: std_logic;
SIGNAL REG_ADDR : std_logic_vector(15 DOWNTO 0);
SIGNAL REG_DIN  : std_logic_vector(15 DOWNTO 0);
SIGNAL REG_DOUT : std_logic_vector(15 DOWNTO 0);
SIGNAL REG_RDEN : std_logic;
SIGNAL REG_WREN : std_logic;
SIGNAL SPARE_DIR: std_logic_vector(11 DOWNTO 0);
SIGNAL SPARE_IN : std_logic_vector(11 DOWNTO 0);
SIGNAL SPARE_OUT: std_logic_vector(11 DOWNTO 0);
SIGNAL USR_ACCESS  : std_logic;


-- Component Declarations
COMPONENT GEMReadout
PORT (
nLBRES: IN  std_logic ;-- Async Reset (active low)
LCLK  : IN  std_logic ;-- Local Bus Clock
--*********************************************
-- REGISTER INTERFACE
--*********************************************
REG_WREN : IN  std_logic ;-- Write pulse (active high)
REG_RDEN : IN  std_logic ;-- Read  pulse (active high)
REG_ADDR : IN  std_logic_vector (15 DOWNTO 0); -- Register address
```

```
   REG_DIN  : IN  std_logic_vector (15 DOWNTO 0); -- Data from CAEN
Local Bus
   REG_DOUT : OUT std_logic_vector (15 DOWNTO 0); -- Data TOCAEN
Local Bus
   USR_ACCESS  : IN  std_logic ;-- Current register access is
   -- at user address space(Active high)
   --***********************************************
   -- V1495 Front Panel Ports (PORT A,B,C,G)
   --***********************************************
   A  : IN  std_logic_vector (31 DOWNTO 0); -- In A (32 x LVDS/ECL)
   B  : IN  std_logic_vector (31 DOWNTO 0); -- In B (32 x LVDS/ECL)
   C  : OUT std_logic_vector (31 DOWNTO 0); -- Out C (32 x LVDS)
   SELG  : OUT std_logic ;-- Output Level Select (NIM/TTL)
   nOEG  : OUT std_logic ;-- Output Enable
   GOUT  : OUT std_logic_vector (1 DOWNTO 0);  -- Out G - LEMO (2 x
NIM/TTL)
   GIN: IN  std_logic_vector (1 DOWNTO 0);  -- In G - LEMO (2 x NIM/
TTL)
   --***********************************************
   -- A395x MEZZANINES INTERFACES (PORT D,E,F)
   --***********************************************
   -- Expansion Mezzanine Identifier:
   -- x_IDCODE :
   -- 000 : A395A (32 x IN LVDS/ECL)
   -- 001 : A395B (32 x OUT LVDS)
   -- 010 : A395C (32 x OUT ECL)
   -- 011 : A395D (8  x IN/OUT NIM/TTL)

   -- Expansion Mezzanine Port Signal Standard Select
   -- x_LEV :
   -- 0=>TTL,1=>NIM

   -- Expansion Mezzanine Port Direction
   -- x_DIR :
   -- 0=>OUT,1=>IN

   -- In/Out D (I/O Expansion)
   IDD: IN  std_logic_vector ( 2 DOWNTO 0); -- D slot mezzanine Identifier
   SELD  : OUT std_logic ;-- D slot Port Signal Standard Select
   nOED  : OUT std_logic ;-- D slot Port Direction
   D  : IN  std_logic_vector (31 DOWNTO 0); -- D slot Data In  Bus
   -- In/Out E (I/O Expansion)
   IDE: IN  std_logic_vector ( 2 DOWNTO 0); -- E slot mezzanine Identifier
```

```
   SELE  : OUT std_logic ;-- E slot Port Signal Standard Select
   nOEE  : OUT std_logic ;-- E slot Port Direction
   E  : OUT std_logic_vector (31 DOWNTO 0); -- E slot Data In  Bus
   -- In/Out F (I/O Expansion)
   IDF: IN  std_logic_vector ( 2 DOWNTO 0); -- F slot mezzanine Identifier
   SELF  : OUT std_logic ;-- F slot Port Signal Standard Select
   nOEF  : OUT std_logic ;-- F slot Port Direction
   F  : IN  std_logic_vector (31 DOWNTO 0); -- F slot Data Out Bus
   --************************************************
   -- DELAY LINES
   --************************************************
   -- PDL = Programmable Delay Lines  (Step = 0.25ns / FSR = 64ns)
   -- DLO = Delay Line Oscillator  (Half Period ~ 10 ns)
   -- 3D3428 PDL (PROGRAMMABLE DELAY LINE) CONFIGURATION
   PDL_WR: OUT std_logic ;-- Write Enable
   PDL_SEL  : OUT std_logic ;-- PDL Selection (0=>PDL0, 1=>PDL1)
   PDL_READ : IN  std_logic_vector ( 7 DOWNTO 0); -- Read Data
   PDL_WRITE: OUT std_logic_vector ( 7 DOWNTO 0); -- Write Data
   PDL_DIR  : OUT std_logic ;-- Direction (0=>Write, 1=>Read)
   -- DELAY I/O
   PDL0_OUT : IN  std_logic ;-- Signal from PDL0 Output
   PDL1_OUT : IN  std_logic ;-- Signal from PDL1 Output
   DLO0_OUT : IN  std_logic ;-- Signal from DLO0 Output
   DLO1_OUT : IN  std_logic ;-- Signal from DLO1 Output
   PDL0_IN  : OUT std_logic ;-- Signal TOPDL0 Input
   PDL1_IN  : OUT std_logic ;-- Signal TOPDL1 Input
   DLO0_GATE: OUT std_logic ;-- DLO0 Gate (active high)
   DLO1_GATE: OUT std_logic ;-- DLO1 Gate (active high)
   --************************************************
   -- SPARE PORTS
   --************************************************
   SPARE_OUT: OUT std_logic_vector (11 DOWNTO 0); -- SPARE Data
Out
   SPARE_IN : IN  std_logic_vector (11 DOWNTO 0); -- SPARE Data In
   SPARE_DIR: OUT std_logic_vector (11 DOWNTO 0); -- SPARE Direction
(0 => OUT, 1 => IN)
   --************************************************
   -- LED
   --************************************************
   RED_PULSE: OUT std_logic ;-- REDLed Pulse (active high)
   GREEN_PULSE : OUT std_logic  -- GREEN Led Pulse (active high)
   );
   END COMPONENT;
```

```
COMPONENT spare_if
PORT (
SPARE_DIR : IN  std_logic_vector (11 DOWNTO 0);
SPARE_OUT : IN  std_logic_vector (11 DOWNTO 0);
SPARE_IN  : OUT std_logic_vector (11 DOWNTO 0);
SPARE  : INOUT  std_logic_vector (11 DOWNTO 0)
);
END COMPONENT;
COMPONENT tristate_if
PORT (
DDLY_OE : IN  std_logic;
DDLY_OUT: IN  std_logic_vector ( 7 DOWNTO 0);
FPGA_DIR: IN  std_logic_vector ( 3 DOWNTO 0);
FPGA_OUT: IN  std_logic_vector ( 3 DOWNTO 0);
LAD_OE  : IN  std_logic;
LAD_OUT : IN  std_logic_vector (15 DOWNTO 0);
DDLY_IN : OUT std_logic_vector ( 7 DOWNTO 0);
FPGA_IN : OUT std_logic_vector ( 3 DOWNTO 0);
LAD_IN  : OUT std_logic_vector (15 DOWNTO 0);
DDLY : INOUT  std_logic_vector ( 7 DOWNTO 0);
FPGA : INOUT  std_logic_vector ( 3 DOWNTO 0);
LAD  : INOUT  std_logic_vector (15 DOWNTO 0)
);
END COMPONENT;
COMPONENT v1495usr_hal
PORT (
DDLY_IN  : IN  std_logic_vector (7 DOWNTO 0);
DLO0_GATE: IN  std_logic ;
DLO1_GATE: IN  std_logic ;
FPGA_IN  : IN  std_logic_vector ( 3 DOWNTO 0);
GREEN_PULSE : IN  std_logic ;
LAD_IN: IN  std_logic_vector (15 DOWNTO 0);
LCLK  : IN  std_logic ;
PDL0_IN  : IN  std_logic ;
PDL1_IN  : IN  std_logic ;
PDL_DIR  : IN  std_logic ;
PDL_SEL  : IN  std_logic ;
PDL_WR: IN  std_logic ;
PDL_WRITE: IN  std_logic_vector ( 7 DOWNTO 0);
PULSE : IN  std_logic_vector (3 DOWNTO 0);
RED_PULSE: IN  std_logic ;
REG_DOUT : IN  std_logic_vector (15 DOWNTO 0);
WnR: IN  std_logic ;
```

```
nADS  : IN  std_logic ;
nBLAST: IN  std_logic ;
nLRESET  : IN  std_logic ;
DDLY_OE  : OUT std_logic ;
DDLY_OUT : OUT std_logic_vector (7 DOWNTO 0);
DIRDDLY  : OUT std_logic ;
DLO0_OUT : OUT std_logic ;
DLO1_OUT : OUT std_logic ;
FPGA_DIR : OUT std_logic_vector ( 3 DOWNTO 0);
FPGA_OUT : OUT std_logic_vector ( 3 DOWNTO 0);
LAD_OE: OUT std_logic ;
LAD_OUT  : OUT std_logic_vector (15 DOWNTO 0);
PDL0_OUT : OUT std_logic ;
PDL1_OUT : OUT std_logic ;
PDL_READ : OUT std_logic_vector ( 7 DOWNTO 0);
REG_ADDR : OUT std_logic_vector (15 DOWNTO 0);
REG_DIN  : OUT std_logic_vector (15 DOWNTO 0);
REG_RDEN : OUT std_logic ;
REG_WREN : OUT std_logic ;
START : OUT std_logic_vector (1 DOWNTO 0);
USR_ACCESS  : OUT std_logic ;
WR_DLY0  : OUT std_logic ;
WR_DLY1  : OUT std_logic ;
nINT  : OUT std_logic ;
nLEDG : OUT std_logic ;
nLEDR : OUT std_logic ;
nOEDDLY0 : OUT std_logic ;
nOEDDLY1 : OUT std_logic ;
nREADY: OUT std_logic ;
nSTART: OUT std_logic_vector (3 DOWNTO 2)
);
END COMPONENT;


BEGIN

-- Instance port mappings.
I0 : GEMReadout
PORT MAP (
nLBRES=> nLRESET,
LCLK  => LCLK,
REG_WREN => REG_WREN,
REG_RDEN => REG_RDEN,
```

```
REG_ADDR => REG_ADDR,
REG_DIN  => REG_DIN,
REG_DOUT => REG_DOUT,
USR_ACCESS  => USR_ACCESS,
A  => A,
B  => B,
C  => C,
SELG  => SELG,
nOEG  => nOEG,
GOUT  => GOUT,
GIN=> GIN,
IDD=> IDD,
SELD  => SELD,
nOED  => nOED,
D  => D,
IDE=> IDE,
SELE  => SELE,
nOEE  => nOEE,
E  => E,
IDF=> IDF,
SELF  => SELF,
nOEF  => nOEF,
F  => F,
PDL_WR=> PDL_WR,
PDL_SEL  => PDL_SEL,
PDL_READ => PDL_READ,
PDL_WRITE=> PDL_WRITE,
PDL_DIR  => PDL_DIR,
PDL0_OUT => PDL0_OUT,
PDL1_OUT => PDL1_OUT,
DLO0_OUT => DLO0_OUT,
DLO1_OUT => DLO1_OUT,
PDL0_IN  => PDL0_IN,
PDL1_IN  => PDL1_IN,
DLO0_GATE=> DLO0_GATE,
DLO1_GATE=> DLO1_GATE,
SPARE_OUT=> SPARE_OUT,
SPARE_IN => SPARE_IN,
SPARE_DIR=> SPARE_DIR,
RED_PULSE=> RED_PULSE,
GREEN_PULSE => GREEN_PULSE
);
I2 : spare_if
```

```
PORT MAP (
SPARE_OUT => SPARE_OUT,
SPARE_IN  => SPARE_IN,
SPARE_DIR => SPARE_DIR,
SPARE  => SPARE
);
I3 : tristate_if
PORT MAP (
LAD_IN  => LAD_IN,
LAD_OE  => LAD_OE,
LAD_OUT => LAD_OUT,
DDLY_OE => DDLY_OE,
DDLY_OUT=> DDLY_OUT,
DDLY_IN => DDLY_IN,
FPGA_IN => FPGA_IN,
FPGA_OUT=> FPGA_OUT,
FPGA_DIR=> FPGA_DIR,
DDLY => DDLY,
LAD  => LAD,
FPGA => FPGA
);

I1 : v1495usr_hal
PORT MAP (
DDLY_IN  => DDLY_IN,
DLO0_GATE=> DLO0_GATE,
DLO1_GATE=> DLO1_GATE,
FPGA_IN  => FPGA_IN,
GREEN_PULSE => GREEN_PULSE,
LAD_IN=> LAD_IN,
LCLK  => LCLK,
PDL0_IN  => PDL0_IN,
PDL1_IN  => PDL1_IN,
PDL_DIR  => PDL_DIR,
PDL_SEL  => PDL_SEL,
PDL_WR=> PDL_WR,
PDL_WRITE=> PDL_WRITE,
PULSE => PULSE,
RED_PULSE=> RED_PULSE,
REG_DOUT => REG_DOUT,
WnR=> WnR,
nADS  => nADS,
nBLAST=> nBLAST,
```

```
nLRESET  => nLRESET,
DDLY_OE  => DDLY_OE,
DDLY_OUT => DDLY_OUT,
DIRDDLY  => DIRDDLY,
DLO0_OUT => DLO0_OUT,
DLO1_OUT => DLO1_OUT,
FPGA_DIR => FPGA_DIR,
FPGA_OUT => FPGA_OUT,
LAD_OE=> LAD_OE,
LAD_OUT  => LAD_OUT,
PDL0_OUT => PDL0_OUT,
PDL1_OUT => PDL1_OUT,
PDL_READ => PDL_READ,
REG_ADDR => REG_ADDR,
REG_DIN  => REG_DIN,
REG_RDEN => REG_RDEN,
REG_WREN => REG_WREN,
START => START,
USR_ACCESS  => USR_ACCESS,
WR_DLY0  => WR_DLY0,
WR_DLY1  => WR_DLY1,
nINT  => nINT,
nLEDG => nLEDG,
nLEDR => nLEDR,
nOEDDLY0 => nOEDDLY0,
nOEDDLY1 => nOEDDLY1,
nREADY=> nREADY,
nSTART=> nSTART
);

END struct;
```

tristate_if_rtl.vhd

```
-- ************************************************************
****************
-- Company:        CAEN SpA - Viareggio - Italy
-- Model:          V1495 -  Multipurpose Programmable Trigger Unit
-- FPGA Proj. Name: V1495USR_DEMO
-- Device:         ALTERA EP1C4F400C6
-- Author:         Luca Colombini
-- Date:           02-03-2006
```

```
-- ------------------------------------------------------------------
-- Module:          TRISTATE_IF
-- Description:     Three-state buffers on bidirectional bus.
-- ********************************************************************
***************

-- ###################################################################
###############
-- Revision History:
-- ###################################################################
###############

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY tristate_if IS
  PORT(
    --************************************************
    -- DATA & CONTROLS (FROM V1495HAL)
    --************************************************
    -- LOCAL BUS DATA BUS
    LAD_IN      : OUT   std_logic_vector (15 DOWNTO 0);
    LAD_OE      : IN    std_logic;
    LAD_OUT     : IN    std_logic_vector (15 DOWNTO 0);
    -- PDL DELAY LINES PARALLEL PROGRAMMING DATA BUS
    DDLY_OE     : IN    std_logic;
    DDLY_OUT    : IN    std_logic_vector ( 7 DOWNTO 0);
    DDLY_IN     : OUT   std_logic_vector ( 7 DOWNTO 0);
    -- FPGA LINK
    FPGA_IN     : OUT   std_logic_vector ( 3 downto 0);
    FPGA_OUT    : IN    std_logic_vector ( 3 downto 0);
    FPGA_DIR    : IN    std_logic_vector ( 3 downto 0);
    --************************************************
    -- BIDIR PORTS
    --************************************************
    DDLY        : INOUT  std_logic_vector ( 7 DOWNTO 0);
    LAD         : INOUT  std_logic_vector (15 DOWNTO 0);
    FPGA        : INOUT  std_logic_vector ( 3 downto 0)
  );
END tristate_if;

--
```

```
ARCHITECTURE rtl OF tristate_if IS
BEGIN

 LAD_IN    <= LAD;
 DDLY_IN   <= DDLY;
 FPGA_IN   <= FPGA;

 process(FPGA_DIR, FPGA_OUT)
 begin
   for i in 0 to 3 loop
     if FPGA_DIR(i) = '1' then
       FPGA(i)    <= FPGA_OUT(i);
     else
       FPGA(i)    <= 'Z';
     end if;
    end loop;
 end process;


 LAD  <= LAD_OUT    when LAD_OE    = '1' else (others => 'Z');
 DDLY <= DDLY_OUT   when DDLY_OE   = '1' else (others => 'Z');

END rtl;
```

spare_if_rtl.vhd

```
   -- ************************************************************
****************
-- Company:        CAEN SpA - Viareggio - Italy
-- Model:          V1495 -  Multipurpose Programmable Trigger Unit
-- FPGA Proj. Name: V1495USR_DEMO
-- Device:         ALTERA EP1C4F400C6
-- Author:         Luca Colombini
-- Date:           02-03-2006
-- ------------------------------------------------------------------------
-- Module:         SPARE_IF
-- Description:    Interface entity to the Spare pins.
--                 It implements three-state buffering of spare pins.
-- ************************************************************
**************
```

```
-- ##################################################################
###############
-- Revision History:
-- ##################################################################
###############

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY spare_if IS
  PORT(

    --************************************************
    -- SPARE PORTS (USER SIDE)
    --************************************************
    SPARE_OUT    : IN   std_logic_vector(11 downto 0);
    SPARE_IN     : OUT  std_logic_vector(11 downto 0);
    SPARE_DIR    : IN   std_logic_vector(11 downto 0);
    --************************************************
    -- SPARE PORTS (I/O PAD)
    --************************************************
    SPARE       : INOUT std_logic_vector(11 downto 0)
  );
END spare_if;

--
ARCHITECTURE rtl OF spare_if IS
BEGIN

  process(SPARE_DIR, SPARE_OUT)
  begin
    for i in 0 to 11 loop
      if SPARE_DIR(i) = '0' then
        SPARE(i)    <= SPARE_OUT(i);
      else
        SPARE(i)    <= 'Z';
      end if;
    end loop;
  end process;

  SPARE_IN <= SPARE;
```

```
END rtl;
```

GEMReadout.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_misc.all;  -- Use OR_REDUCE function

use work.v1495pkg.all;

entity GEMReadout is
      port(
nLBRES     : IN    std_logic ;                  -- Async Reset (active low)
LCLK       : IN    std_logic ;                  -- Local Bus Clock
--***********************************************
-- REGISTER INTERFACE
--***********************************************
REG_WREN   : IN    std_logic ;                  -- Write pulse (active high)
REG_RDEN   : IN    std_logic ;                  -- Read  pulse (active high)
REG_ADDR   : IN    std_logic_vector (15 DOWNTO 0); -- Register
address
REG_DIN    : IN    std_logic_vector (15 DOWNTO 0); -- Data from
CAEN Local Bus
REG_DOUT   : OUT   std_logic_vector (15 DOWNTO 0); -- Data TO
CAEN Local Bus
USR_ACCESS : IN    std_logic ;                  -- Current register access is
-- at user address space(Active high)
--***********************************************
-- V1495 Front Panel Ports (PORT A,B,C,G)
--***********************************************
A          : IN    std_logic_vector (31 DOWNTO 0); -- In A (32 x LVDS/
ECL)
B          : IN    std_logic_vector (31 DOWNTO 0); -- In B (32 x LVDS/
ECL)
C          : OUT   std_logic_vector (31 DOWNTO 0); -- Out C (32 x LVDS)
SELG       : OUT   std_logic ;                  -- Output Level Select (NIM/
TTL)
nOEG       : OUT   std_logic ;                  -- Output Enable
GOUT       : OUT   std_logic_vector (1 DOWNTO 0);  -- Out G - LEMO
```

```
(2 x NIM/TTL)
   GIN        : IN     std_logic_vector (1 DOWNTO 0);  -- In G - LEMO (2 x
NIM/TTL)
   --***********************************************
   -- A395x MEZZANINES INTERFACES (PORT D,E,F)
   --***********************************************
   -- Expansion Mezzanine Identifier:
   -- x_IDCODE :
   -- 000 : A395A (32 x IN LVDS/ECL)
   -- 001 : A395B (32 x OUT LVDS)
   -- 010 : A395C (32 x OUT ECL)
   -- 011 : A395D (8  x IN/OUT NIM/TTL)

   -- Expansion Mezzanine Port Signal Standard Select
   -- x_LEV :
   --    0=>TTL,1=>NIM

   -- Expansion Mezzanine Port Direction
   -- x_DIR :
   --    0=>OUT,1=>IN

   -- In/Out D (I/O Expansion)
   IDD        : IN     std_logic_vector ( 2 DOWNTO 0); -- D slot mezzanine
Identifier
   SELD       : OUT    std_logic ;                  -- D slot Port Signal Standard
Select
   nOED       : OUT    std_logic ;                  -- D slot Port Direction
   D          : IN     std_logic_vector (31 DOWNTO 0); -- D slot Data In  Bus
   -- In/Out E (I/O Expansion)
   IDE        : IN     std_logic_vector ( 2 DOWNTO 0); -- E slot mezzanine
Identifier
   SELE       : OUT    std_logic ;                  -- E slot Port Signal Standard
Select
   nOEE       : OUT    std_logic ;                  -- E slot Port Direction
   E          : OUT    std_logic_vector (31 DOWNTO 0); -- E slot Data In  Bus
   -- In/Out F (I/O Expansion)
   IDF        : IN     std_logic_vector ( 2 DOWNTO 0); -- F slot mezzanine
Identifier
   SELF       : OUT    std_logic ;                  -- F slot Port Signal Standard
Select
   nOEF       : OUT    std_logic ;                  -- F slot Port Direction
   F          : IN     std_logic_vector (31 DOWNTO 0); -- F slot Data Out Bus
   --***********************************************
```

```
   -- DELAY LINES
   --************************************************
   -- PDL = Programmable Delay Lines  (Step = 0.25ns / FSR = 64ns)
   -- DLO = Delay Line Oscillator     (Half Period ~ 10 ns)
   -- 3D3428 PDL (PROGRAMMABLE DELAY LINE) CONFIGURATION
   PDL_WR      : OUT    std_logic ;                 -- Write Enable
   PDL_SEL     : OUT    std_logic ;                 -- PDL Selection (0=>PDL0,
1=>PDL1)
   PDL_READ    : IN     std_logic_vector ( 7 DOWNTO 0); -- Read Data
   PDL_WRITE   : OUT    std_logic_vector ( 7 DOWNTO 0); -- Write Data
   PDL_DIR     : OUT    std_logic ;                 -- Direction (0=>Write,
1=>Read)
   -- DELAY I/O
   PDL0_OUT    : IN     std_logic ;                -- Signal from PDL0 Output
   PDL1_OUT    : IN     std_logic ;                -- Signal from PDL1 Output
   DLO0_OUT    : IN     std_logic ;                 -- Signal from DLO0 Output
   DLO1_OUT    : IN     std_logic ;                 -- Signal from DLO1 Output
   PDL0_IN     : OUT    std_logic ;                -- Signal TO   PDL0 Input
   PDL1_IN     : OUT    std_logic ;                -- Signal TO   PDL1 Input
   DLO0_GATE   : OUT    std_logic ;                   -- DLO0 Gate (active high)
   DLO1_GATE   : OUT    std_logic ;                   -- DLO1 Gate (active high)
   --************************************************
   -- SPARE PORTS
   --************************************************
   SPARE_OUT   : OUT    std_logic_vector (11 DOWNTO 0); -- SPARE Data
Out
   SPARE_IN    : IN     std_logic_vector (11 DOWNTO 0); -- SPARE Data In
   SPARE_DIR   : OUT    std_logic_vector (11 DOWNTO 0); -- SPARE
Direction (0 => OUT, 1 => IN)
   --************************************************
   -- LED
   --************************************************
   RED_PULSE   : OUT    std_logic ;                -- RED   Led Pulse (active
high)
   GREEN_PULSE : OUT    std_logic                    -- GREEN Led Pulse
(active high)
         );
   end GEMReadout;

   architecture Synthesis of GEMReadout is
         component PLLBlock
   port(
   areset: in std_logic := '0';
```

```
inclk0: in std_logic := '0';
c0    : out std_logic;
c1    : out std_logic;
locked: out std_logic
      );
      end component;

      component GEMTrigger is
port(-- LCLK Synchronous Section
LCLK        : in std_logic;
SOFT_TRIGGER   : in std_logic;
CALIB_TRIGGER   : in std_logic;
SOFT_TRIG_WORD       : in std_logic_vector(2 downto 0);
HARD_TRIG_WORD       : in std_logic_vector(2 downto 0);

-- CLK Synchronous Section
CLK: in std_logic;
RESET       : in std_logic;

HARD_TRIGGER : in std_logic;

T1: out std_logic;
DEBUG_OUT: out std_logic;
DEBUG_OUT2: out std_logic
      );
      end component;

      component GEMTxChannel is
port(-- LCLK Synchronous Section
LCLK        : in std_logic;

GEM_TX_WORD_0       : in std_logic_vector(15 downto 0);
GEM_TX_WORD_1       : in std_logic_vector(15 downto 0);
GEM_TX_WORD_2       : in std_logic_vector(15 downto 0);
GEM_TX_WORD_3       : in std_logic_vector(15 downto 0);
GEM_TX_WORD_4       : in std_logic_vector(15 downto 0);
GEM_TX_WORD_5       : in std_logic_vector(15 downto 0);
GEM_TX_WORD_6       : in std_logic_vector(15 downto 0);
GEM_TX_WORD_7       : in std_logic_vector(15 downto 0);
GEM_TX_WORD_8       : in std_logic_vector(15 downto 0);
GEM_TX_WORD_9       : in std_logic_vector(15 downto 0);
GEM_TX_WORD_A       : in std_logic_vector(15 downto 0);
GEM_TX_WORD_B       : in std_logic_vector(15 downto 0);
```

```
    GEM_TX_START  : in std_logic;

    -- CLK Synchronous Section
    CLK: in std_logic;

    TX_EXT_EN: in std_logic;
    HARD_TRIGGER : in std_logic;
    DATA         : out std_logic;
    DATA_VALID: out std_logic
          );
          end component;

          component GEMRxChannel is
    port(-- LCLK Synchronous Section
    LCLK         : in std_logic;

    RD_EVENT_SIZE : in std_logic;
    EVENT_SIZE: out std_logic_vector(3 downto 0);
    EVENT_COUNT: out std_logic_vector(5 downto 0);

    RD_EVENT_DATA          : in std_logic;
    EVENT_DATA: out std_logic_vector(15 downto 0);
    EVENT_DATA_SIZE        : out std_logic_vector(9 downto 0);

    GEM_EVENTS_SENT      : out std_logic_vector(31 downto 0);

    -- CLK Synchronous Section
    CLK: in std_logic;
    RESET        : in std_logic;

    DATA         : in std_logic;
    DATA_VALID: in std_logic
          );
          end component;

          constant V_REVISION          : std_logic_vector(15 downto 0) :=
x”0201”;

          signal HEART_BEAT_CNT: std_logic_vector(25 downto 0);

          signal REG_RDEN_Q          : std_logic;
          signal REG_WREN_Q          : std_logic;
```

```
        signal T_LWORD: std_logic_vector(15 downto 0);

        signal PLL_LOCK: std_logic;
        signal PLLCLK: std_logic;
        signal PLLCLK_90    : std_logic;

        signal LBRES: std_logic;
        signal SRESET: std_logic_vector(2 downto 0);
        signal SYS_RESET    : std_logic;
        signal RESET: std_logic;

        signal REG_RESET    : std_logic;

        -- GEM Breakout Box A Event Buffer Signals
        signal GEMA0_RD_EVENT_SIZE, GEMA1_RD_EVENT_SIZE,
GEMA2_RD_EVENT_SIZE: std_logic;
        signal GEMA3_RD_EVENT_SIZE, GEMA4_RD_EVENT_SIZE,
GEMA5_RD_EVENT_SIZE: std_logic;

        signal GEMA0_EVENT_SIZE, GEMA1_EVENT_SIZE, GEMA2_
EVENT_SIZE : std_logic_vector(3 downto 0);
        signal GEMA3_EVENT_SIZE, GEMA4_EVENT_SIZE, GEMA5_
EVENT_SIZE : std_logic_vector(3 downto 0);

        signal GEMA0_EVENT_COUNT, GEMA1_EVENT_COUNT,
GEMA2_EVENT_COUNT: std_logic_vector(5 downto 0);
        signal GEMA3_EVENT_COUNT, GEMA4_EVENT_COUNT,
GEMA5_EVENT_COUNT: std_logic_vector(5 downto 0);

        signal GEMA0_RD_EVENT_DATA, GEMA1_RD_EVENT_DATA,
GEMA2_RD_EVENT_DATA: std_logic;
        signal GEMA3_RD_EVENT_DATA, GEMA4_RD_EVENT_DATA,
GEMA5_RD_EVENT_DATA: std_logic;

        signal GEMA0_EVENT_DATA, GEMA1_EVENT_DATA, GEMA2_
EVENT_DATA          : std_logic_vector(15 downto 0);
        signal GEMA3_EVENT_DATA, GEMA4_EVENT_DATA, GEMA5_
EVENT_DATA          : std_logic_vector(15 downto 0);

        signal GEMA0_EVENT_DATA_SIZE, GEMA1_EVENT_DATA_
SIZE, GEMA2_EVENT_DATA_SIZE         : std_logic_vector(9 downto 0);
        signal GEMA3_EVENT_DATA_SIZE, GEMA4_EVENT_DATA_
SIZE, GEMA5_EVENT_DATA_SIZE         : std_logic_vector(9 downto 0);
```

```
        signal GEMA0_EVENTS_SENT, GEMA1_EVENTS_SENT,
GEMA2_EVENTS_SENT: std_logic_vector(31 downto 0);
        signal GEMA3_EVENTS_SENT, GEMA4_EVENTS_SENT,
GEMA5_EVENTS_SENT: std_logic_vector(31 downto 0);

        -- GEM Breakout Box A Input Signals
        signal GEMA0_DATA, GEMA1_DATA, GEMA2_DATA   : std_logic;
        signal GEMA3_DATA, GEMA4_DATA, GEMA5_DATA   : std_logic;
        signal GEMA0_DATA_VALID, GEMA1_DATA_VALID, GEMA2_
DATA_VALID          : std_logic;
        signal GEMA3_DATA_VALID, GEMA4_DATA_VALID, GEMA5_
DATA_VALID          : std_logic;

        -- GEM Breakout Box B Event Buffer Signals
        signal GEMB0_RD_EVENT_SIZE, GEMB1_RD_EVENT_SIZE,
GEMB2_RD_EVENT_SIZE: std_logic;
        signal GEMB3_RD_EVENT_SIZE, GEMB4_RD_EVENT_SIZE,
GEMB5_RD_EVENT_SIZE: std_logic;

        signal GEMB0_EVENT_SIZE, GEMB1_EVENT_SIZE, GEMB2_
EVENT_SIZE: std_logic_vector(3 downto 0);
        signal GEMB3_EVENT_SIZE, GEMB4_EVENT_SIZE, GEMB5_
EVENT_SIZE: std_logic_vector(3 downto 0);

        signal GEMB0_EVENT_COUNT, GEMB1_EVENT_COUNT,
GEMB2_EVENT_COUNT: std_logic_vector(5 downto 0);
        signal GEMB3_EVENT_COUNT, GEMB4_EVENT_COUNT,
GEMB5_EVENT_COUNT: std_logic_vector(5 downto 0);

        signal GEMB0_RD_EVENT_DATA, GEMB1_RD_EVENT_DATA,
GEMB2_RD_EVENT_DATA: std_logic;
        signal GEMB3_RD_EVENT_DATA, GEMB4_RD_EVENT_DATA,
GEMB5_RD_EVENT_DATA: std_logic;

        signal GEMB0_EVENT_DATA, GEMB1_EVENT_DATA, GEMB2_
EVENT_DATA          : std_logic_vector(15 downto 0);
        signal GEMB3_EVENT_DATA, GEMB4_EVENT_DATA, GEMB5_
EVENT_DATA          : std_logic_vector(15 downto 0);

        signal GEMB0_EVENT_DATA_SIZE, GEMB1_EVENT_DATA_
SIZE, GEMB2_EVENT_DATA_SIZE         : std_logic_vector(9 downto 0);
        signal GEMB3_EVENT_DATA_SIZE, GEMB4_EVENT_DATA_
```

```
SIZE, GEMB5_EVENT_DATA_SIZE         : std_logic_vector(9 downto 0);

        signal GEMB0_EVENTS_SENT, GEMB1_EVENTS_SENT,
GEMB2_EVENTS_SENT: std_logic_vector(31 downto 0);
        signal GEMB3_EVENTS_SENT, GEMB4_EVENTS_SENT,
GEMB5_EVENTS_SENT: std_logic_vector(31 downto 0);

        -- GEM Breakout Box B Input Signals
        signal GEMB0_DATA, GEMB1_DATA, GEMB2_DATA    : std_logic;
        signal GEMB3_DATA, GEMB4_DATA, GEMB5_DATA    : std_logic;
        signal GEMB0_DATA_VALID, GEMB1_DATA_VALID, GEMB2_
DATA_VALID          : std_logic;
        signal GEMB3_DATA_VALID, GEMB4_DATA_VALID, GEMB5_
DATA_VALID          : std_logic;

        -- GEM Simulated DATA/DATAVALID Signals
        signal GEM_TX_WORD_0, GEM_TX_WORD_1, GEM_TX_
WORD_2, GEM_TX_WORD_3       : std_logic_vector(15 downto 0);
        signal GEM_TX_WORD_4, GEM_TX_WORD_5, GEM_TX_
WORD_6, GEM_TX_WORD_7       : std_logic_vector(15 downto 0);
        signal GEM_TX_WORD_8, GEM_TX_WORD_9, GEM_TX_
WORD_A, GEM_TX_WORD_B      : std_logic_vector(15 downto 0);
        signal GEM_TX_START       : std_logic;
        signal GEM_TX_DATA        : std_logic;
        signal GEM_TX_DATA_VALID      : std_logic;
        signal GEM_TX_START_EXT_EN  : std_logic;

        -- GEM Trigger Signals
        signal SOFT_TRIGGER       : std_logic;
        signal HARD_TRIGGER       : std_logic;
        signal SOFT_TRIG_WORD: std_logic_vector(2 downto 0);
        signal HARD_TRIG_WORD: std_logic_vector(2 downto 0);
        signal T1         : std_logic;

        -- Calibration Signals
        signal CALIB_TRIGGER: std_logic;
        signal CALIB_TRIGGER_EN: std_logic;
        signal CALIB_TRIG_SIZE: std_logic_vector(15 downto 0);
        signal CALIB_TRIGGER_LCLK      : std_logic;

        signal DEBUG_OUT : std_logic;
        signal DEBUG_OUT2          : std_logic;
```

```
begin

    RED_PULSE <= SYS_RESET;
    GREEN_PULSE <= HEART_BEAT_CNT(25);
    SYS_RESET <= not nLBRES or not PLL_LOCK;
    -- SYS_RESET <= '0';

    LBRES <= not nLBRES;

    nOED <= '0';  -- Not used
    nOEE <= '0';  -- Not used
    nOEF <= '1';  -- Inputs
    nOEG <= '1';  -- Inputs
    SELD <= '0';  -- Not used
    SELE <= '0';  -- Not used
    SELF <= '0';  -- NIM
    SELG <= '1';  -- TTL Level on G

    ------------ Unused IO ------------
    --A(1 downto 0) <= (others => '0');
    --A(15 downto 14) <= (others => '0');
    --A(16 downto 13) <= (others => '0');
    --A(24 downto 18) <= (others => '0');
    --A(31 downto 26) <= (others => '0');
    C(1 downto 0) <= (others => 'Z');
    C(14 downto 15) <= (others => 'Z');
    --C(31 downto 30) <= (others => '0');
    --C(15 downto 12) <= (others => '0');
    --warrenC(15 downto 12) <= (others => HARD_TRIGGER);
    --E(31 downto 28) <= (others => '0');
    --E(15 downto 12) <= (others => '0');

    GOUT <= "00";          -- Not used
    SPARE_OUT <= (others => '0');
    SPARE_DIR <= (others => '1');
    PDL_WR <= '0';
    PDL_SEL <= '0';
    PDL_WRITE <= (others => '0');
    PDL_DIR <= '0';
    PDL0_IN <= '0';
    PDL1_IN <= '0';
    DLO0_GATE <= '0';
```

```
        DLO1_GATE <= '0';
        ----------------------------------

        process(PLLCLK, RESET)
        begin
if RESET='1' then
        GEMA0_DATA_VALID <= A(6); GEMA0_DATA <= A(7);
        GEMA1_DATA_VALID <= A(4); GEMA1_DATA <= A(5);
        GEMA2_DATA_VALID <= A(2); GEMA2_DATA <= A(3);
        GEMA3_DATA_VALID <= A(13); GEMA3_DATA <= A(12);
        GEMA4_DATA_VALID <= A(11); GEMA4_DATA <= A(10);
        GEMA5_DATA_VALID <= A(9); GEMA5_DATA <= A(8);

        --GEMA0_DATA_VALID <= A(29); GEMA0_DATA <= A(30);
        --GEMA1_DATA_VALID <= A(27); GEMA1_DATA <= A(28);
        --GEMA2_DATA_VALID <= A(25); GEMA2_DATA <= A(26);
        --GEMA3_DATA_VALID <= A(23); GEMA3_DATA <= A(24);
        --GEMA4_DATA_VALID <= A(21); GEMA4_DATA <= A(22);
        --GEMA5_DATA_VALID <= A(19); GEMA5_DATA <= A(20);
elsif rising_edge(PLLCLK) then
        -- GEM Breakout Box A Input Signals
        GEMA0_DATA_VALID <= A(6); GEMA0_DATA <= A(7);
        GEMA1_DATA_VALID <= A(4); GEMA1_DATA <= A(5);
        GEMA2_DATA_VALID <= A(2); GEMA2_DATA <= A(3);
        GEMA3_DATA_VALID <= A(13); GEMA3_DATA <= A(12);
        GEMA4_DATA_VALID <= A(11); GEMA4_DATA <= A(10);
        GEMA5_DATA_VALID <= A(9); GEMA5_DATA <= A(8);

        --GEMA0_DATA_VALID <= A(29); GEMA0_DATA <= A(30);
        --GEMA1_DATA_VALID <= A(27); GEMA1_DATA <= A(28);
        --GEMA2_DATA_VALID <= A(25); GEMA2_DATA <= A(26);
        --GEMA3_DATA_VALID <= A(23); GEMA3_DATA <= A(24);
        --GEMA4_DATA_VALID <= A(21); GEMA4_DATA <= A(22);
        --GEMA5_DATA_VALID <= A(19); GEMA5_DATA <= A(20);

        -- GEM Breakout Box B Input Signals
        --GEMB0_DATA_VALID <= A(16); GEMB0_DATA <= A(17);
        --GEMB1_DATA_VALID <= A(18); GEMB1_DATA <= A(19);
        --GEMB2_DATA_VALID <= A(20); GEMB2_DATA <= A(21);
        --GEMB3_DATA_VALID <= A(22); GEMB3_DATA <= A(23);
        --GEMB4_DATA_VALID <= A(24); GEMB4_DATA <= A(25);
        --GEMB5_DATA_VALID <= A(26); GEMB5_DATA <= A(27);
```

```
        -- GEM Output Test Signals A
        --E(0) <= GEM_TX_DATA_VALID; E(1) <= GEM_TX_DATA;
        --E(2) <= GEM_TX_DATA_VALID; E(3) <= GEM_TX_DATA;
        --E(4) <= GEM_TX_DATA_VALID; E(5) <= GEM_TX_DATA;
        --E(6) <= GEM_TX_DATA_VALID; E(7) <= GEM_TX_DATA;
        --E(8) <= GEM_TX_DATA_VALID; E(9) <= GEM_TX_DATA;
        --E(10) <= GEM_TX_DATA_VALID; E(11) <= GEM_TX_DATA;

        -- GEM Output Test Signals B
        --E(16) <= GEM_TX_DATA_VALID; E(17) <= GEM_TX_DATA;
        --E(18) <= GEM_TX_DATA_VALID; E(19) <= GEM_TX_DATA;
        --E(20) <= GEM_TX_DATA_VALID; E(21) <= GEM_TX_DATA;
        --E(22) <= GEM_TX_DATA_VALID; E(23) <= GEM_TX_DATA;
        --E(24) <= GEM_TX_DATA_VALID; E(25) <= GEM_TX_DATA;
        --E(26) <= GEM_TX_DATA_VALID; E(27) <= GEM_TX_DATA;
end if;
        end process;

        process(PLLCLK_90)
        begin
if rising_edge(PLLCLK_90) then
        -- GEM Breakout Box A T1 Signals
        C(3) <= T1;
        C(5) <= T1;
        C(7) <= T1;
        C(8) <= T1;
        C(10) <= T1;
        C(12) <= T1;
        --C(15) <= T1;

        -- GEM Debug Testing Signals
        C(16) <= T1;
           end if;
        end process;

        process(PLLCLK)
        begin
if rising_edge(PLLCLK) then
        -- GEM Breakout Box B T1 Signals

        -- External Trigger Input

        if CALIB_TRIGGER_LCLK = '0' then
```

```
HARD_TRIGGER <= GIN(1);
        end if;
        -- For Trigger Pass-through
        --T1 <= HARD_TRIGGER;

end if;
        end process;

        -- GEM Breakout Box A CLK Signals
        C(2) <= PLLCLK_90;
        C(4) <= PLLCLK_90;
        C(6) <= PLLCLK_90;
        C(9) <= PLLCLK_90;
        C(11) <= PLLCLK_90;
        C(13) <= PLLCLK_90;
        --C(14) <= PLLCLK_90;

        --GEM Debug Testing Signals
        C(17) <= PLLCLK_90;
        C(18) <= GEMA0_DATA;
        C(19) <= GEMA0_DATA_VALID;
        C(20) <= GEMA1_DATA;
        C(21) <= GEMA1_DATA_VALID;
        C(22) <= GEMA2_DATA;
        C(23) <= GEMA2_DATA_VALID;
        C(24) <= GEMA3_DATA;
        C(25) <= GEMA3_DATA_VALID;
        C(26) <= GEMA4_DATA;
        C(27) <= GEMA4_DATA_VALID;
        C(28) <= GEMA5_DATA;
        C(29) <= GEMA5_DATA_VALID;
        C(30) <= CALIB_TRIGGER;
        C(31) <= CALIB_TRIGGER_LCLK;
        --C(7) <= 'Z';
        --C(9) <= 'Z';
        --C(11) <= PLLCLK_90;
        --C(10) <= HARD_TRIGGER;
        --C(11) <= HARD_TRIGGER;

        -- GEM Breakout Box B CLK Signals
        --C(17) <= PLLCLK_90;
        --C(19) <= PLLCLK_90;
        --C(21) <= PLLCLK_90;
```

```
        --C(23) <= PLLCLK_90;
        --C(25) <= PLLCLK_90;
        --C(27) <= PLLCLK_90;

        PLL_0: PLLBlock
port map(    areset => LBRES,
        inclk0 => GIN(0),
        c0 => PLLCLK,
        c1 => PLLCLK_90,
        locked => PLL_LOCK
        );

        --PLL_LOCK <= '1';
    --PLLCLK <= GIN(0);
    --PLLCLK_90 <= not PLLCLK;

        GEMTrigger_0: GEMTrigger
port map(    LCLK  => LCLK,
        SOFT_TRIGGER      => SOFT_TRIGGER,
        CALIB_TRIGGER   => CALIB_TRIGGER,
        SOFT_TRIG_WORD => SOFT_TRIG_WORD,
        HARD_TRIG_WORD          => HARD_TRIG_WORD,
        CLK=> PLLCLK,
        RESET           => RESET,
        HARD_TRIGGER     => HARD_TRIGGER,
        T1=> T1,
        DEBUG_OUT=> DEBUG_OUT,
        DEBUG_OUT2=> DEBUG_OUT2
        );

        GEMTxChannel_0: GEMTxChannel
port map(    LCLK  => LCLK,
        GEM_TX_WORD_0 => GEM_TX_WORD_0,
        GEM_TX_WORD_1 => GEM_TX_WORD_1,
        GEM_TX_WORD_2 => GEM_TX_WORD_2,
        GEM_TX_WORD_3 => GEM_TX_WORD_3,
        GEM_TX_WORD_4 => GEM_TX_WORD_4,
        GEM_TX_WORD_5 => GEM_TX_WORD_5,
        GEM_TX_WORD_6 => GEM_TX_WORD_6,
        GEM_TX_WORD_7 => GEM_TX_WORD_7,
        GEM_TX_WORD_8 => GEM_TX_WORD_8,
        GEM_TX_WORD_9 => GEM_TX_WORD_9,
        GEM_TX_WORD_A => GEM_TX_WORD_A,
```

```
        GEM_TX_WORD_B => GEM_TX_WORD_B,
        GEM_TX_START     => GEM_TX_START,
        CLK=> PLLCLK,
        TX_EXT_EN=> GEM_TX_START_EXT_EN,
        HARD_TRIGGER     => HARD_TRIGGER,
        DATA  => GEM_TX_DATA,
        DATA_VALID=> GEM_TX_DATA_VALID
        );

        GEMRxChannel_A0: GEMRxChannel
    port map(    LCLK  => LCLK,
        RD_EVENT_SIZE     => GEMA0_RD_EVENT_SIZE,
        EVENT_SIZE=> GEMA0_EVENT_SIZE,
        EVENT_COUNT=> GEMA0_EVENT_COUNT,
        RD_EVENT_DATA   => GEMA0_RD_EVENT_DATA,
        EVENT_DATA=> GEMA0_EVENT_DATA,
        EVENT_DATA_SIZE=> GEMA0_EVENT_DATA_SIZE,
        GEM_EVENTS_SENT         => GEMA0_EVENTS_SENT,
        CLK=> PLLCLK,
        RESET          => RESET,
        DATA  => GEMA0_DATA,
        DATA_VALID=> GEMA0_DATA_VALID
        );

        GEMRxChannel_A1: GEMRxChannel
    port map(    LCLK  => LCLK,
        RD_EVENT_SIZE     => GEMA1_RD_EVENT_SIZE,
        EVENT_SIZE=> GEMA1_EVENT_SIZE,
        EVENT_COUNT=> GEMA1_EVENT_COUNT,
        RD_EVENT_DATA   => GEMA1_RD_EVENT_DATA,
        EVENT_DATA=> GEMA1_EVENT_DATA,
        EVENT_DATA_SIZE=> GEMA1_EVENT_DATA_SIZE,
        GEM_EVENTS_SENT         => GEMA1_EVENTS_SENT,
        CLK=> PLLCLK,
        RESET          => RESET,
        DATA  => GEMA1_DATA,
        DATA_VALID=> GEMA1_DATA_VALID
        );

        GEMRxChannel_A2: GEMRxChannel
    port map(    LCLK  => LCLK,
        RD_EVENT_SIZE     => GEMA2_RD_EVENT_SIZE,
        EVENT_SIZE=> GEMA2_EVENT_SIZE,
```

```
        EVENT_COUNT=> GEMA2_EVENT_COUNT,
        RD_EVENT_DATA   => GEMA2_RD_EVENT_DATA,
        EVENT_DATA=> GEMA2_EVENT_DATA,
        EVENT_DATA_SIZE=> GEMA2_EVENT_DATA_SIZE,
        GEM_EVENTS_SENT         => GEMA2_EVENTS_SENT,
        CLK=> PLLCLK,
        RESET           => RESET,
        DATA  => GEMA2_DATA,
        DATA_VALID=> GEMA2_DATA_VALID
        );

        GEMRxChannel_A3: GEMRxChannel
    port map(    LCLK  => LCLK,
        RD_EVENT_SIZE     => GEMA3_RD_EVENT_SIZE,
        EVENT_SIZE=> GEMA3_EVENT_SIZE,
        EVENT_COUNT=> GEMA3_EVENT_COUNT,
        RD_EVENT_DATA   => GEMA3_RD_EVENT_DATA,
        EVENT_DATA=> GEMA3_EVENT_DATA,
        EVENT_DATA_SIZE=> GEMA3_EVENT_DATA_SIZE,
        GEM_EVENTS_SENT         => GEMA3_EVENTS_SENT,
        CLK=> PLLCLK,
        RESET           => RESET,
        DATA  => GEMA3_DATA,
        DATA_VALID=> GEMA3_DATA_VALID
        );

        GEMRxChannel_A4: GEMRxChannel
    port map(    LCLK  => LCLK,
        RD_EVENT_SIZE     => GEMA4_RD_EVENT_SIZE,
        EVENT_SIZE=> GEMA4_EVENT_SIZE,
        EVENT_COUNT=> GEMA4_EVENT_COUNT,
        RD_EVENT_DATA   => GEMA4_RD_EVENT_DATA,
        EVENT_DATA=> GEMA4_EVENT_DATA,
        EVENT_DATA_SIZE=> GEMA4_EVENT_DATA_SIZE,
        GEM_EVENTS_SENT         => GEMA4_EVENTS_SENT,
        CLK=> PLLCLK,
        RESET           => RESET,
        DATA  => GEMA4_DATA,
        DATA_VALID=> GEMA4_DATA_VALID
        );

        GEMRxChannel_A5: GEMRxChannel
    port map(    LCLK  => LCLK,
```

```
        RD_EVENT_SIZE     => GEMA5_RD_EVENT_SIZE,
        EVENT_SIZE=> GEMA5_EVENT_SIZE,
        EVENT_COUNT=> GEMA5_EVENT_COUNT,
        RD_EVENT_DATA   => GEMA5_RD_EVENT_DATA,
        EVENT_DATA=> GEMA5_EVENT_DATA,
        EVENT_DATA_SIZE=> GEMA5_EVENT_DATA_SIZE,
        GEM_EVENTS_SENT         => GEMA5_EVENTS_SENT,
        CLK=> PLLCLK,
        RESET           => RESET,
        DATA  => GEMA5_DATA,
        DATA_VALID=> GEMA5_DATA_VALID
        );

        GEMRxChannel_B0: GEMRxChannel
    port map(    LCLK  => LCLK,
        RD_EVENT_SIZE     => GEMB0_RD_EVENT_SIZE,
        EVENT_SIZE=> GEMB0_EVENT_SIZE,
        EVENT_COUNT=> GEMB0_EVENT_COUNT,
        RD_EVENT_DATA   => GEMB0_RD_EVENT_DATA,
        EVENT_DATA=> GEMB0_EVENT_DATA,
        EVENT_DATA_SIZE=> GEMB0_EVENT_DATA_SIZE,
        GEM_EVENTS_SENT         => GEMB0_EVENTS_SENT,
        CLK=> PLLCLK,
        RESET           => RESET,
        DATA  => GEMB0_DATA,
        DATA_VALID=> GEMB0_DATA_VALID
        );

        GEMRxChannel_B1: GEMRxChannel
    port map(    LCLK  => LCLK,
        RD_EVENT_SIZE     => GEMB1_RD_EVENT_SIZE,
        EVENT_SIZE=> GEMB1_EVENT_SIZE,
        EVENT_COUNT=> GEMB1_EVENT_COUNT,
        RD_EVENT_DATA   => GEMB1_RD_EVENT_DATA,
        EVENT_DATA=> GEMB1_EVENT_DATA,
        EVENT_DATA_SIZE=> GEMB1_EVENT_DATA_SIZE,
        GEM_EVENTS_SENT         => GEMB1_EVENTS_SENT,
        CLK=> PLLCLK,
        RESET           => RESET,
        DATA  => GEMB1_DATA,
        DATA_VALID=> GEMB1_DATA_VALID
        );
```

```
        GEMRxChannel_B2: GEMRxChannel
    port map(    LCLK  => LCLK,
        RD_EVENT_SIZE     => GEMB2_RD_EVENT_SIZE,
        EVENT_SIZE=> GEMB2_EVENT_SIZE,
        EVENT_COUNT=> GEMB2_EVENT_COUNT,
        RD_EVENT_DATA   => GEMB2_RD_EVENT_DATA,
        EVENT_DATA=> GEMB2_EVENT_DATA,
        EVENT_DATA_SIZE=> GEMB2_EVENT_DATA_SIZE,
        GEM_EVENTS_SENT         => GEMB2_EVENTS_SENT,
        CLK=> PLLCLK,
        RESET           => RESET,
        DATA  => GEMB2_DATA,
        DATA_VALID=> GEMB2_DATA_VALID
        );

        GEMRxChannel_B3: GEMRxChannel
    port map(    LCLK  => LCLK,
        RD_EVENT_SIZE     => GEMB3_RD_EVENT_SIZE,
        EVENT_SIZE=> GEMB3_EVENT_SIZE,
        EVENT_COUNT=> GEMB3_EVENT_COUNT,
        RD_EVENT_DATA   => GEMB3_RD_EVENT_DATA,
        EVENT_DATA=> GEMB3_EVENT_DATA,
        EVENT_DATA_SIZE=> GEMB3_EVENT_DATA_SIZE,
        GEM_EVENTS_SENT         => GEMB3_EVENTS_SENT,
        CLK=> PLLCLK,
        RESET           => RESET,
        DATA  => GEMB3_DATA,
        DATA_VALID=> GEMB3_DATA_VALID
        );

        GEMRxChannel_B4: GEMRxChannel
    port map(    LCLK  => LCLK,
        RD_EVENT_SIZE     => GEMB4_RD_EVENT_SIZE,
        EVENT_SIZE=> GEMB4_EVENT_SIZE,
        EVENT_COUNT=> GEMB4_EVENT_COUNT,
        RD_EVENT_DATA   => GEMB4_RD_EVENT_DATA,
        EVENT_DATA=> GEMB4_EVENT_DATA,
        EVENT_DATA_SIZE=> GEMB4_EVENT_DATA_SIZE,
        GEM_EVENTS_SENT         => GEMB4_EVENTS_SENT,
        CLK=> PLLCLK,
        RESET           => RESET,
        DATA  => GEMB4_DATA,
        DATA_VALID=> GEMB4_DATA_VALID
```

```
        );

        GEMRxChannel_B5: GEMRxChannel
port map(    LCLK  => LCLK,
        RD_EVENT_SIZE     => GEMB5_RD_EVENT_SIZE,
        EVENT_SIZE=> GEMB5_EVENT_SIZE,
        EVENT_COUNT=> GEMB5_EVENT_COUNT,
        RD_EVENT_DATA   => GEMB5_RD_EVENT_DATA,
        EVENT_DATA=> GEMB5_EVENT_DATA,
        EVENT_DATA_SIZE=> GEMB5_EVENT_DATA_SIZE,
        GEM_EVENTS_SENT         => GEMB5_EVENTS_SENT,
        CLK=> PLLCLK,
        RESET           => RESET,
        DATA  => GEMB5_DATA,
        DATA_VALID=> GEMB5_DATA_VALID
        );

        process(LCLK, SYS_RESET)
        begin
if SYS_RESET = '1' then
        SRESET <= "111";
        RESET <= '1';
elsif rising_edge(LCLK) then
        SRESET <= REG_RESET & SRESET(2 downto 1);
        RESET <= SRESET(0);
end if;
        end process;

        process(LCLK)
        begin
if rising_edge(LCLK) then
        REG_WREN_Q <= REG_WREN;
        REG_RDEN_Q <= REG_RDEN;
end if;
        end process;

        process(LCLK, nLBRES)
        begin
if nLBRES = '0' then
        HEART_BEAT_CNT <= (others => '0');
elsif rising_edge(LCLK) then
        HEART_BEAT_CNT <= HEART_BEAT_CNT + 1;
end if;
```

```
        end process;

        process(LCLK, nLBRES)
        begin
if nLBRES = '0' then
        CALIB_TRIGGER_LCLK <= '0';
elsif rising_edge(LCLK) then
        if CALIB_TRIGGER_EN = '1' then
CALIB_TRIGGER_LCLK <= '1';
        elsif CALIB_TRIGGER_LCLK = '1' then
if CALIB_TRIG_SIZE < x"FFFF" then
        CALIB_TRIGGER <= '1';
        CALIB_TRIGGER_LCLK <= '0';
else
        CALIB_TRIGGER <= GIN(1);
end if;
        else
CALIB_TRIGGER <= '0';
        end if;
end if;
        end process;

        -- WRITE REGISTERS
        P_WREG : process(LCLK, nLBRES)
        begin
if (nLBRES = '0') then
        GEM_TX_WORD_0 <= (others => '0');
        GEM_TX_WORD_1 <= (others => '0');
        GEM_TX_WORD_2 <= (others => '0');
        GEM_TX_WORD_3 <= (others => '0');
        GEM_TX_WORD_4 <= (others => '0');
        GEM_TX_WORD_5 <= (others => '0');
        GEM_TX_WORD_6 <= (others => '0');
        GEM_TX_WORD_7 <= (others => '0');
        GEM_TX_WORD_8 <= (others => '0');
        GEM_TX_WORD_9 <= (others => '0');
        GEM_TX_WORD_A <= (others => '0');
        GEM_TX_WORD_B <= (others => '0');
        GEM_TX_START <= '0';
        SOFT_TRIGGER <= '0';
        SOFT_TRIG_WORD <= (others => '0');
        HARD_TRIG_WORD <= "100";
        REG_RESET <= '0';
```

```
        GEM_TX_START_EXT_EN <= '0';
        CALIB_TRIGGER_EN <= '0';
    elsif LCLK'event and LCLK = '1' then

        if REG_WREN_Q = '0' then
    GEM_TX_START <= '0';
    SOFT_TRIGGER <= '0';
    REG_RESET <= '0';
    CALIB_TRIGGER_EN <= '0';
        end if;

        if REG_WREN_Q = '0' and REG_WREN = '1' and USR_ACCESS =
'1' then
    case REG_ADDR is
        when A_RESET        => REG_RESET <= '1';
        when A_GEM_CALIB_START      => CALIB_TRIG_SIZE <=
REG_DIN; CALIB_TRIGGER_EN <= '1';
        when A_GEM_TX_START=> GEM_TX_START_EXT_EN <= REG_
DIN(1); GEM_TX_START <= REG_DIN(0);
        when A_GEM_SOFT_TRIG  => SOFT_TRIGGER <= '1';
        when A_GEM_TRIG_WORD        => HARD_TRIG_WORD <=
REG_DIN(5 downto 3); SOFT_TRIG_WORD <= REG_DIN(2 downto 0);
        when A_GEM_TX_WORD_0        => GEM_TX_WORD_0 <=
REG_DIN;
        when A_GEM_TX_WORD_1        => GEM_TX_WORD_1 <=
REG_DIN;
        when A_GEM_TX_WORD_2        => GEM_TX_WORD_2 <=
REG_DIN;
        when A_GEM_TX_WORD_3        => GEM_TX_WORD_3 <=
REG_DIN;
        when A_GEM_TX_WORD_4        => GEM_TX_WORD_4 <=
REG_DIN;
        when A_GEM_TX_WORD_5        => GEM_TX_WORD_5 <=
REG_DIN;
        when A_GEM_TX_WORD_6        => GEM_TX_WORD_6 <=
REG_DIN;
        when A_GEM_TX_WORD_7        => GEM_TX_WORD_7 <=
REG_DIN;
        when A_GEM_TX_WORD_8        => GEM_TX_WORD_8 <=
REG_DIN;
        when A_GEM_TX_WORD_9        => GEM_TX_WORD_9 <=
REG_DIN;
        when A_GEM_TX_WORD_A        => GEM_TX_WORD_A <=
```

```
REG_DIN;
        when A_GEM_TX_WORD_B        => GEM_TX_WORD_B <=
REG_DIN;

        when others =>          null;
   end case;
        end if;
   end if;
        end process;

        -- READ REGISTERS
        P_RREG: process(LCLK, nLBRES)
        begin
   if (nLBRES = '0') then
        REG_DOUT <= (others => '0');
        T_LWORD <= (others => '0');
        GEMA0_RD_EVENT_SIZE <= '0';
        GEMA1_RD_EVENT_SIZE <= '0';
        GEMA2_RD_EVENT_SIZE <= '0';
        GEMA3_RD_EVENT_SIZE <= '0';
        GEMA4_RD_EVENT_SIZE <= '0';
        GEMA5_RD_EVENT_SIZE <= '0';
        GEMB0_RD_EVENT_SIZE <= '0';
        GEMB1_RD_EVENT_SIZE <= '0';
        GEMB2_RD_EVENT_SIZE <= '0';
        GEMB3_RD_EVENT_SIZE <= '0';
        GEMB4_RD_EVENT_SIZE <= '0';
        GEMB5_RD_EVENT_SIZE <= '0';
        GEMA0_RD_EVENT_DATA <= '0';
        GEMA1_RD_EVENT_DATA <= '0';
        GEMA2_RD_EVENT_DATA <= '0';
        GEMA3_RD_EVENT_DATA <= '0';
        GEMA4_RD_EVENT_DATA <= '0';
        GEMA5_RD_EVENT_DATA <= '0';
        GEMB0_RD_EVENT_DATA <= '0';
        GEMB1_RD_EVENT_DATA <= '0';
        GEMB2_RD_EVENT_DATA <= '0';
        GEMB3_RD_EVENT_DATA <= '0';
        GEMB4_RD_EVENT_DATA <= '0';
        GEMB5_RD_EVENT_DATA <= '0';
   elsif LCLK'event and LCLK = '1' then
        REG_DOUT   <= (others => '0');
```

```
        GEMA0_RD_EVENT_SIZE <= '0';
        GEMA1_RD_EVENT_SIZE <= '0';
        GEMA2_RD_EVENT_SIZE <= '0';
        GEMA3_RD_EVENT_SIZE <= '0';
        GEMA4_RD_EVENT_SIZE <= '0';
        GEMA5_RD_EVENT_SIZE <= '0';

        GEMB0_RD_EVENT_SIZE <= '0';
        GEMB1_RD_EVENT_SIZE <= '0';
        GEMB2_RD_EVENT_SIZE <= '0';
        GEMB3_RD_EVENT_SIZE <= '0';
        GEMB4_RD_EVENT_SIZE <= '0';
        GEMB5_RD_EVENT_SIZE <= '0';

        GEMA0_RD_EVENT_DATA <= '0';
        GEMA1_RD_EVENT_DATA <= '0';
        GEMA2_RD_EVENT_DATA <= '0';
        GEMA3_RD_EVENT_DATA <= '0';
        GEMA4_RD_EVENT_DATA <= '0';
        GEMA5_RD_EVENT_DATA <= '0';

        GEMB0_RD_EVENT_DATA <= '0';
        GEMB1_RD_EVENT_DATA <= '0';
        GEMB2_RD_EVENT_DATA <= '0';
        GEMB3_RD_EVENT_DATA <= '0';
        GEMB4_RD_EVENT_DATA <= '0';
        GEMB5_RD_EVENT_DATA <= '0';

        if REG_RDEN_Q = '0' and REG_RDEN = '1' and USR_ACCESS =
'1' then
        --if REG_RDEN = '1' and USR_ACCESS = '1' then
   if REG_ADDR(14) = '1' then
        case REG_ADDR(15 downto 4) is
   when A_GEMA0_EVENTDATA(15 downto 4)   => REG_DOUT <=
GEMA0_EVENT_DATA; GEMA0_RD_EVENT_DATA <= '1';
   when A_GEMA1_EVENTDATA(15 downto 4)   => REG_DOUT <=
GEMA1_EVENT_DATA; GEMA1_RD_EVENT_DATA <= '1';
   when A_GEMA2_EVENTDATA(15 downto 4)   => REG_DOUT <=
GEMA2_EVENT_DATA; GEMA2_RD_EVENT_DATA <= '1';
   when A_GEMA3_EVENTDATA(15 downto 4)   => REG_DOUT <=
GEMA3_EVENT_DATA; GEMA3_RD_EVENT_DATA <= '1';
   when A_GEMA4_EVENTDATA(15 downto 4)   => REG_DOUT <=
GEMA4_EVENT_DATA; GEMA4_RD_EVENT_DATA <= '1';
```

```
   when A_GEMA5_EVENTDATA(15 downto 4)    => REG_DOUT <=
GEMA5_EVENT_DATA; GEMA5_RD_EVENT_DATA <= '1';
   when A_GEMB0_EVENTDATA(15 downto 4)    => REG_DOUT <=
GEMB0_EVENT_DATA; GEMB0_RD_EVENT_DATA <= '1';
   when A_GEMB1_EVENTDATA(15 downto 4)    => REG_DOUT <=
GEMB1_EVENT_DATA; GEMB1_RD_EVENT_DATA <= '1';
   when A_GEMB2_EVENTDATA(15 downto 4)    => REG_DOUT <=
GEMB2_EVENT_DATA; GEMB2_RD_EVENT_DATA <= '1';
   when A_GEMB3_EVENTDATA(15 downto 4)    => REG_DOUT <=
GEMB3_EVENT_DATA; GEMB3_RD_EVENT_DATA <= '1';
   when A_GEMB4_EVENTDATA(15 downto 4)    => REG_DOUT <=
GEMB4_EVENT_DATA; GEMB4_RD_EVENT_DATA <= '1';
   when A_GEMB5_EVENTDATA(15 downto 4)    => REG_DOUT <=
GEMB5_EVENT_DATA; GEMB5_RD_EVENT_DATA <= '1';
   when others => REG_DOUT <= (others => '0');
         end case;
   else
         case REG_ADDR is
   when A_BOARDIDS=> REG_DOUT <= "0000000" & IDF & IDE & IDD;
   when A_REVISION=> REG_DOUT <= V_REVISION;
   when A_GEM_CALIB_START=> REG_DOUT <= CALIB_TRIG_SIZE;
   when A_GEM_TRIG_WORD          => REG_DOUT <= "0000000000" &
HARD_TRIG_WORD & SOFT_TRIG_WORD;
   when A_GEMA0_FIFOSIZE=> REG_DOUT <= GEMA0_EVENT_
DATA_SIZE & GEMA0_EVENT_COUNT;
   when A_GEMA1_FIFOSIZE=> REG_DOUT <= GEMA1_EVENT_
DATA_SIZE & GEMA1_EVENT_COUNT;
   when A_GEMA2_FIFOSIZE=> REG_DOUT <= GEMA2_EVENT_
DATA_SIZE & GEMA2_EVENT_COUNT;
   when A_GEMA3_FIFOSIZE=> REG_DOUT <= GEMA3_EVENT_
DATA_SIZE & GEMA3_EVENT_COUNT;
   when A_GEMA4_FIFOSIZE=> REG_DOUT <= GEMA4_EVENT_
DATA_SIZE & GEMA4_EVENT_COUNT;
   when A_GEMA5_FIFOSIZE=> REG_DOUT <= GEMA5_EVENT_
DATA_SIZE & GEMA5_EVENT_COUNT;

   when A_GEMB0_FIFOSIZE=> REG_DOUT <= GEMB0_EVENT_
DATA_SIZE & GEMB0_EVENT_COUNT;
   when A_GEMB1_FIFOSIZE=> REG_DOUT <= GEMB1_EVENT_
DATA_SIZE & GEMB1_EVENT_COUNT;
   when A_GEMB2_FIFOSIZE=> REG_DOUT <= GEMB2_EVENT_
DATA_SIZE & GEMB2_EVENT_COUNT;
   when A_GEMB3_FIFOSIZE=> REG_DOUT <= GEMB3_EVENT_
```

```
DATA_SIZE & GEMB3_EVENT_COUNT;
   when A_GEMB4_FIFOSIZE=> REG_DOUT <= GEMB4_EVENT_
DATA_SIZE & GEMB4_EVENT_COUNT;
   when A_GEMB5_FIFOSIZE=> REG_DOUT <= GEMB5_EVENT_
DATA_SIZE & GEMB5_EVENT_COUNT;

   when A_GEMA0_EVENTSIZE=> REG_DOUT <= x"000" & GEMA0_
EVENT_SIZE; GEMA0_RD_EVENT_SIZE <= '1';
   when A_GEMA1_EVENTSIZE=> REG_DOUT <= x"000" & GEMA1_
EVENT_SIZE; GEMA1_RD_EVENT_SIZE <= '1';
   when A_GEMA2_EVENTSIZE=> REG_DOUT <= x"000" & GEMA2_
EVENT_SIZE; GEMA2_RD_EVENT_SIZE <= '1';
   when A_GEMA3_EVENTSIZE=> REG_DOUT <= x"000" & GEMA3_
EVENT_SIZE; GEMA3_RD_EVENT_SIZE <= '1';
   when A_GEMA4_EVENTSIZE=> REG_DOUT <= x"000" & GEMA4_
EVENT_SIZE; GEMA4_RD_EVENT_SIZE <= '1';
   when A_GEMA5_EVENTSIZE=> REG_DOUT <= x"000" & GEMA5_
EVENT_SIZE; GEMA5_RD_EVENT_SIZE <= '1';

   when A_GEMB0_EVENTSIZE=> REG_DOUT <= x"000" & GEMB0_
EVENT_SIZE; GEMB0_RD_EVENT_SIZE <= '1';
   when A_GEMB1_EVENTSIZE=> REG_DOUT <= x"000" & GEMB1_
EVENT_SIZE; GEMB1_RD_EVENT_SIZE <= '1';
   when A_GEMB2_EVENTSIZE=> REG_DOUT <= x"000" & GEMB2_
EVENT_SIZE; GEMB2_RD_EVENT_SIZE <= '1';
   when A_GEMB3_EVENTSIZE=> REG_DOUT <= x"000" & GEMB3_
EVENT_SIZE; GEMB3_RD_EVENT_SIZE <= '1';
   when A_GEMB4_EVENTSIZE=> REG_DOUT <= x"000" & GEMB4_
EVENT_SIZE; GEMB4_RD_EVENT_SIZE <= '1';
   when A_GEMB5_EVENTSIZE=> REG_DOUT <= x"000" & GEMB5_
EVENT_SIZE; GEMB5_RD_EVENT_SIZE <= '1';

   when A_GEMA0_EVENTS_SENT_H     => REG_DOUT <= GEMA0_
EVENTS_SENT(31 downto 16); T_LWORD <= GEMA0_EVENTS_
SENT(15 downto 0);
   when A_GEMA0_EVENTS_SENT_L     => REG_DOUT <= T_LWORD;
   when A_GEMA1_EVENTS_SENT_H     => REG_DOUT <= GEMA1_
EVENTS_SENT(31 downto 16); T_LWORD <= GEMA1_EVENTS_
SENT(15 downto 0);
   when A_GEMA1_EVENTS_SENT_L     => REG_DOUT <= T_LWORD;
   when A_GEMA2_EVENTS_SENT_H     => REG_DOUT <= GEMA2_
EVENTS_SENT(31 downto 16); T_LWORD <= GEMA2_EVENTS_
SENT(15 downto 0);
```

```
   when A_GEMA2_EVENTS_SENT_L      => REG_DOUT <= T_LWORD;
   when A_GEMA3_EVENTS_SENT_H      => REG_DOUT <= GEMA3_
EVENTS_SENT(31 downto 16); T_LWORD <= GEMA3_EVENTS_
SENT(15 downto 0);
   when A_GEMA3_EVENTS_SENT_L      => REG_DOUT <= T_LWORD;
   when A_GEMA4_EVENTS_SENT_H      => REG_DOUT <= GEMA4_
EVENTS_SENT(31 downto 16); T_LWORD <= GEMA4_EVENTS_
SENT(15 downto 0);
   when A_GEMA4_EVENTS_SENT_L      => REG_DOUT <= T_LWORD;
   when A_GEMA5_EVENTS_SENT_H      => REG_DOUT <= GEMA5_
EVENTS_SENT(31 downto 16); T_LWORD <= GEMA5_EVENTS_
SENT(15 downto 0);
   when A_GEMA5_EVENTS_SENT_L      => REG_DOUT <= T_LWORD;

   when A_GEMB0_EVENTS_SENT_H      => REG_DOUT <= GEMB0_
EVENTS_SENT(31 downto 16); T_LWORD <= GEMB0_EVENTS_
SENT(15 downto 0);
   when A_GEMB0_EVENTS_SENT_L      => REG_DOUT <= T_LWORD;
   when A_GEMB1_EVENTS_SENT_H      => REG_DOUT <= GEMB1_
EVENTS_SENT(31 downto 16); T_LWORD <= GEMB1_EVENTS_
SENT(15 downto 0);
   when A_GEMB1_EVENTS_SENT_L      => REG_DOUT <= T_LWORD;
   when A_GEMB2_EVENTS_SENT_H      => REG_DOUT <= GEMB2_
EVENTS_SENT(31 downto 16); T_LWORD <= GEMB2_EVENTS_
SENT(15 downto 0);
   when A_GEMB2_EVENTS_SENT_L      => REG_DOUT <= T_LWORD;
   when A_GEMB3_EVENTS_SENT_H      => REG_DOUT <= GEMB3_
EVENTS_SENT(31 downto 16); T_LWORD <= GEMB3_EVENTS_
SENT(15 downto 0);
   when A_GEMB3_EVENTS_SENT_L      => REG_DOUT <= T_LWORD;
   when A_GEMB4_EVENTS_SENT_H      => REG_DOUT <= GEMB4_
EVENTS_SENT(31 downto 16); T_LWORD <= GEMB4_EVENTS_
SENT(15 downto 0);
   when A_GEMB4_EVENTS_SENT_L      => REG_DOUT <= T_LWORD;
   when A_GEMB5_EVENTS_SENT_H      => REG_DOUT <= GEMB5_
EVENTS_SENT(31 downto 16); T_LWORD <= GEMB5_EVENTS_
SENT(15 downto 0);
   when A_GEMB5_EVENTS_SENT_L      => REG_DOUT <= T_LWORD;

   when others => REG_DOUT <= x”0000”;
         end case;
   end if;
         end if;
```

```
    end if;
        end process;

    end Synthesis;
```

GEMTrigger.vhd

```
    library ieee;
    use ieee.std_logic_1164.all;
    use ieee.std_logic_arith.all;
    use ieee.std_logic_unsigned.all;
    use ieee.std_logic_misc.all;  -- Use OR_REDUCE function

    use work.v1495pkg.all;

    entity GEMTrigger is
        port(    -- LCLK Synchronous Section
                        LCLK                    : in std_logic;
                        RESET                   : in std_logic;
                        SOFT_TRIGGER    : in std_logic;
                        CALIB_TRIGGER   : in std_logic;
                        SOFT_TRIG_WORD : in std_logic_vector(2 downto
0);
                        HARD_TRIG_WORD          : in std_logic_vector(2
downto 0);

                        -- CLK Synchronous Section
                        CLK                     : in std_logic;
                        HARD_TRIGGER    : in std_logic;

                        T1                      : out std_logic;
                        DEBUG_OUT       : out std_logic;
                        DEBUG_OUT2              : out std_logic
                );
    end GEMTrigger;

    architecture Synthesis of GEMTrigger is
        signal T1_SHIFT_REG             : std_logic_vector(2
downto 0);
        signal SHIFT_REG_EMPTY          : std_logic;

        --signal RESET_UP           : std_logic;
```

```
        signal RESET_CLK_UP             : std_logic;
        signal RESET_CLK_Q1             : std_logic;
        signal RESET_CLK_Q2             : std_logic;
        signal RESET_LCLK_UP            : std_logic;
        signal RESET_LCLK_Q1            : std_logic;
        signal RESET_LCLK_Q2            : std_logic;

        signal HARD_TRIGGER_UP                  : std_logic;
        signal HARD_TRIGGER_Q1                  : std_logic;
        signal HARD_TRIGGER_Q2                  : std_logic;

        signal SOFT_TRIGGER_UP          : std_logic;
        signal SOFT_TRIGGER_Q1          : std_logic;
        signal SOFT_TRIGGER_Q2          : std_logic;

        signal CALIB_LCLK_UP            : std_logic;
        signal CALIB_TRIGGER_UP                 : std_logic;
        signal CALIB_CLK_UP                     : std_logic;
        signal CALIB_TRIG_WORD                  : std_logic_vector(2
downto 0);
        signal CALIB_PULSE_CNT          : std_logic_vector(3 downto 0);

    begin

        process(LCLK)
        begin
                if RESET = '1' then
                        RESET_LCLK_UP <= '1';
                elsif RESET_CLK_UP = '1' then
                        RESET_LCLK_UP <= '0';
                end if;
        end process;

        process(CLK)
        begin
                if rising_edge(CLK) then
                        if RESET_LCLK_UP = '1' then
                                RESET_CLK_UP <= '1';
                        else
                                RESET_CLK_UP <= '0';
                        end if;
                end if;
        end process;
```

```
	process(LCLK)
	begin
		if CALIB_TRIGGER = '1' then
			CALIB_LCLK_UP <= '1';
		elsif CALIB_CLK_UP = '1' then
			CALIB_LCLK_UP <= '0';
		end if;
	end process;

	process(CLK)
	begin
		if rising_edge(CLK) then
			if CALIB_LCLK_UP = '1' then
				CALIB_CLK_UP <= '1';
				CALIB_PULSE_CNT <= "0000";
			elsif CALIB_CLK_UP = '1' AND CALIB_PULSE_
CNT = "0000" then
				CALIB_TRIG_WORD <= "111";
				CALIB_TRIGGER_UP <= '1';
				CALIB_PULSE_CNT <= CALIB_PULSE_
CNT + 1;
			elsif CALIB_CLK_UP = '1' and CALIB_PULSE_CNT
= "0111" then
				CALIB_TRIG_WORD <= "100";
				CALIB_TRIGGER_UP <= '1';
				CALIB_PULSE_CNT <= CALIB_PULSE_
CNT + 1;
			elsif CALIB_CLK_UP = '1' and CALIB_PULSE_CNT
= "1010" then
				CALIB_CLK_UP <= '0';
			elsif CALIB_CLK_UP = '1' then
				CALIB_PULSE_CNT <= CALIB_PULSE_
CNT + 1;
				CALIB_TRIGGER_UP <= '0';
			end if;
		end if;
	end process;

	-- SOFT_TRIGGER_UP <= SOFT_TRIGGER_Q1 and not SOFT_
TRIGGER_Q2;

	process(LCLK)
```

```
	begin
		if rising_edge(LCLK) then
			SOFT_TRIGGER_Q1 <= SOFT_TRIGGER;
			SOFT_TRIGGER_Q2 <= SOFT_TRIGGER_Q1;
		end if;

		if SOFT_TRIGGER_Q1 = '1' and not SOFT_TRIGGER_Q2 =
'1' then
			SOFT_TRIGGER_UP <= '1';
		end if;
	end process;

	HARD_TRIGGER_UP <= HARD_TRIGGER_Q1 and not HARD_
TRIGGER_Q2;

	process(CLK)
	begin
		if rising_edge(CLK) then
			HARD_TRIGGER_Q1 <= HARD_TRIGGER;
			HARD_TRIGGER_Q2 <= HARD_TRIGGER_Q1;
		end if;
	end process;

	SHIFT_REG_EMPTY <= not (T1_SHIFT_REG(0) or T1_SHIFT_
REG(1) or T1_SHIFT_REG(2));

	process(CLK)
	begin
		if rising_edge(CLK) then
			if RESET_LCLK_UP = '1' and SHIFT_REG_EMPTY
= '1' then
				T1_SHIFT_REG <= "110";		-- T1
command to reset VFATs
			elsif CALIB_TRIGGER_UP = '1' AND SHIFT_REG_
EMPTY = '1' then
				T1_SHIFT_REG <= CALIB_TRIG_WORD;
			elsif HARD_TRIGGER_UP = '1' and SHIFT_REG_
EMPTY = '1' then
				T1_SHIFT_REG <= HARD_TRIG_WORD;
			elsif SOFT_TRIGGER_UP = '1' and SHIFT_REG_
EMPTY = '1' then
				T1_SHIFT_REG <= SOFT_TRIG_WORD;
			--	SOFT_TRIGGER_UP <= '0';
```

```
                        else
                                T1_SHIFT_REG <= T1_SHIFT_REG(1 downto
0) & '0';
                        end if;
                end if;
        end process;

        T1 <= T1_SHIFT_REG(2);
        --T1 <= RESET_LCLK_UP;
        DEBUG_OUT <= CALIB_CLK_UP;
        DEBUG_OUT2 <= CALIB_PULSE_CNT(0);

    end Synthesis;v1495usr_pkg.vhd
```

PLLBlock.vhd

```
    -- megafunction wizard: %ALTPLL%
    -- GENERATION: STANDARD
    -- VERSION: WM1.0
    -- MODULE: altpll

    -- ============================================================
========
    -- File Name: PLLBlock.vhd
    -- Megafunction Name(s):
    --                     altpll
    --
    -- Simulation Library Files(s):
    --                     altera_mf
    -- ============================================================
========
    -- ************************************************************
*
    -- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
    --
    -- 7.2 Build 207 03/18/2008 SP 3 SJ Full Version
    -- ************************************************************
*


    --Copyright (C) 1991-2007 Altera Corporation
    --Your use of Altera Corporation's design tools, logic functions
```

```
--and other software and tools, and its AMPP partner logic
--functions, and any output files from any of the foregoing
--(including device programming or simulation files), and any
--associated documentation or information are expressly subject
--to the terms and conditions of the Altera Program License
--Subscription Agreement, Altera MegaCore Function License
--Agreement, or other applicable license agreement, including,
--without limitation, that your use is for the sole purpose of
--programming logic devices manufactured by Altera and sold by
--Altera or its authorized distributors.  Please refer to the
--applicable agreement for further details.


LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY altera_mf;
USE altera_mf.all;

ENTITY PLLBlock IS
	PORT
	(
		areset		: IN STD_LOGIC  := '0';
		inclk0		: IN STD_LOGIC  := '0';
		c0		: OUT STD_LOGIC ;
		c1		: OUT STD_LOGIC ;
		locked		: OUT STD_LOGIC
	);
END PLLBlock;


ARCHITECTURE SYN OF pllblock IS

	SIGNAL sub_wire0	: STD_LOGIC_VECTOR (5 DOWNTO 0);
	SIGNAL sub_wire1	: STD_LOGIC ;
	SIGNAL sub_wire2	: STD_LOGIC ;
	SIGNAL sub_wire3	: STD_LOGIC ;
	SIGNAL sub_wire4	: STD_LOGIC ;
	SIGNAL sub_wire5	: STD_LOGIC_VECTOR (1 DOWNTO 0);
	SIGNAL sub_wire6_bv	: BIT_VECTOR (0 DOWNTO 0);
	SIGNAL sub_wire6	: STD_LOGIC_VECTOR (0 DOWNTO 0);
```

```
COMPONENT altpll
GENERIC (
        clk0_divide_by                  : NATURAL;
        clk0_duty_cycle                 : NATURAL;
        clk0_multiply_by                : NATURAL;
        clk0_phase_shift                : STRING;
        clk1_divide_by                  : NATURAL;
        clk1_duty_cycle                 : NATURAL;
        clk1_multiply_by                : NATURAL;
        clk1_phase_shift                : STRING;
        compensate_clock                : STRING;
        inclk0_input_frequency                  : NATURAL;
        intended_device_family                  : STRING;
        invalid_lock_multiplier                 : NATURAL;
        lpm_hint                : STRING;
        lpm_type                : STRING;
        operation_mode                  : STRING;
        pll_type                : STRING;
        port_activeclock                : STRING;
        port_areset             : STRING;
        port_clkbad0            : STRING;
        port_clkbad1            : STRING;
        port_clkloss            : STRING;
        port_clkswitch          : STRING;
        port_configupdate               : STRING;
        port_fbin               : STRING;
        port_inclk0             : STRING;
        port_inclk1             : STRING;
        port_locked             : STRING;
        port_pfdena             : STRING;
        port_phasecounterselect                 : STRING;
        port_phasedone                  : STRING;
        port_phasestep                  : STRING;
        port_phaseupdown                : STRING;
        port_pllena             : STRING;
        port_scanaclr           : STRING;
        port_scanclk            : STRING;
        port_scanclkena                 : STRING;
        port_scandata           : STRING;
        port_scandataout                : STRING;
        port_scandone           : STRING;
        port_scanread           : STRING;
```

```
            port_scanwrite          : STRING;
            port_clk0               : STRING;
            port_clk1               : STRING;
            port_clk3               : STRING;
            port_clk4               : STRING;
            port_clk5               : STRING;
            port_clkena0            : STRING;
            port_clkena1            : STRING;
            port_clkena3            : STRING;
            port_clkena4            : STRING;
            port_clkena5            : STRING;
            port_extclk0            : STRING;
            port_extclk1            : STRING;
            port_extclk2            : STRING;
            port_extclk3            : STRING;
            valid_lock_multiplier           : NATURAL
    );
    PORT (
                inclk   : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
                locked  : OUT STD_LOGIC ;
                areset  : IN STD_LOGIC ;
                clk     : OUT STD_LOGIC_VECTOR (5 DOWNTO 0)
    );
    END COMPONENT;

BEGIN
    sub_wire6_bv(0 DOWNTO 0) <= "0";
    sub_wire6   <= To_stdlogicvector(sub_wire6_bv);
    sub_wire2   <= sub_wire0(1);
    sub_wire1   <= sub_wire0(0);
    c0   <= sub_wire1;
    c1   <= sub_wire2;
    locked   <= sub_wire3;
    sub_wire4   <= inclk0;
    sub_wire5   <= sub_wire6(0 DOWNTO 0) & sub_wire4;

    altpll_component : altpll
    GENERIC MAP (
            clk0_divide_by => 1,
            clk0_duty_cycle => 50,
            clk0_multiply_by => 1,
            clk0_phase_shift => "0",
            clk1_divide_by => 1,
```

```
clk1_duty_cycle => 50,
clk1_multiply_by => 1,
clk1_phase_shift => “12500”,
compensate_clock => “CLK0”,
inclk0_input_frequency => 25000,
intended_device_family => “Cyclone”,
invalid_lock_multiplier => 5,
lpm_hint => “CBX_MODULE_PREFIX=PLLBlock”,
lpm_type => “altpll”,
operation_mode => “NORMAL”,
pll_type => “AUTO”,
port_activeclock => “PORT_UNUSED”,
port_areset => “PORT_USED”,
port_clkbad0 => “PORT_UNUSED”,
port_clkbad1 => “PORT_UNUSED”,
port_clkloss => “PORT_UNUSED”,
port_clkswitch => “PORT_UNUSED”,
port_configupdate => “PORT_UNUSED”,
port_fbin => “PORT_UNUSED”,
port_inclk0 => “PORT_USED”,
port_inclk1 => “PORT_UNUSED”,
port_locked => “PORT_USED”,
port_pfdena => “PORT_UNUSED”,
port_phasecounterselect => “PORT_UNUSED”,
port_phasedone => “PORT_UNUSED”,
port_phasestep => “PORT_UNUSED”,
port_phaseupdown => “PORT_UNUSED”,
port_pllena => “PORT_UNUSED”,
port_scanaclr => “PORT_UNUSED”,
port_scanclk => “PORT_UNUSED”,
port_scanclkena => “PORT_UNUSED”,
port_scandata => “PORT_UNUSED”,
port_scandataout => “PORT_UNUSED”,
port_scandone => “PORT_UNUSED”,
port_scanread => “PORT_UNUSED”,
port_scanwrite => “PORT_UNUSED”,
port_clk0 => “PORT_USED”,
port_clk1 => “PORT_USED”,
port_clk3 => “PORT_UNUSED”,
port_clk4 => “PORT_UNUSED”,
port_clk5 => “PORT_UNUSED”,
port_clkena0 => “PORT_UNUSED”,
port_clkena1 => “PORT_UNUSED”,
```

```
                port_clkena3 => “PORT_UNUSED”,
                port_clkena4 => “PORT_UNUSED”,
                port_clkena5 => “PORT_UNUSED”,
                port_extclk0 => “PORT_UNUSED”,
                port_extclk1 => “PORT_UNUSED”,
                port_extclk2 => “PORT_UNUSED”,
                port_extclk3 => “PORT_UNUSED”,
                valid_lock_multiplier => 1
        )
        PORT MAP (
                inclk => sub_wire5,
                areset => areset,
                clk => sub_wire0,
                locked => sub_wire3
        );



END SYN;
```

GEMRxEventDataFIFO.vhd

```
-- megafunction wizard: %FIFO%
-- GENERATION: STANDARD
-- VERSION: WM1.0
-- MODULE: dcfifo

-- ============================================================
========
-- File Name: GEMRxEventDataFIFO.vhd
-- Megafunction Name(s):
--                      dcfifo
--
-- Simulation Library Files(s):
--                      altera_mf
-- ============================================================
========
-- ************************************************************
*
-- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
--
-- 7.2 Build 207 03/18/2008 SP 3 SJ Full Version
-- ************************************************************
```

*

```
   --Copyright (C) 1991-2007 Altera Corporation
   --Your use of Altera Corporation's design tools, logic functions
   --and other software and tools, and its AMPP partner logic
   --functions, and any output files from any of the foregoing
   --(including device programming or simulation files), and any
   --associated documentation or information are expressly subject
   --to the terms and conditions of the Altera Program License
   --Subscription Agreement, Altera MegaCore Function License
   --Agreement, or other applicable license agreement, including,
   --without limitation, that your use is for the sole purpose of
   --programming logic devices manufactured by Altera and sold by
   --Altera or its authorized distributors.  Please refer to the
   --applicable agreement for further details.


   LIBRARY ieee;
   USE ieee.std_logic_1164.all;

   LIBRARY altera_mf;
   USE altera_mf.all;

   ENTITY GEMRxEventDataFIFO IS
         PORT
         (
                  aclr              : IN STD_LOGIC  := '0';
                  data              : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
                  rdclk             : IN STD_LOGIC ;
                  rdreq             : IN STD_LOGIC ;
                  wrclk             : IN STD_LOGIC ;
                  wrreq             : IN STD_LOGIC ;
                  q                 : OUT STD_LOGIC_VECTOR (15 DOWNTO
0);
                  rdempty                 : OUT STD_LOGIC ;
                  rdusedw                 : OUT STD_LOGIC_VECTOR (9
DOWNTO 0);
                  wrfull            : OUT STD_LOGIC
         );
   END GEMRxEventDataFIFO;
```

```
ARCHITECTURE SYN OF gemrxeventdatafifo IS

        SIGNAL sub_wire0    : STD_LOGIC ;
        SIGNAL sub_wire1    : STD_LOGIC ;
        SIGNAL sub_wire2    : STD_LOGIC_VECTOR (15 DOWNTO 0);
        SIGNAL sub_wire3    : STD_LOGIC_VECTOR (9 DOWNTO 0);



        COMPONENT dcfifo
        GENERIC (
                add_ram_output_register         : STRING;
                clocks_are_synchronized         : STRING;
                intended_device_family          : STRING;
                lpm_hint                : STRING;
                lpm_numwords                    : NATURAL;
                lpm_showahead                   : STRING;
                lpm_type                : STRING;
                lpm_width               : NATURAL;
                lpm_widthu              : NATURAL;
                overflow_checking               : STRING;
                underflow_checking              : STRING;
                use_eab                 : STRING
        );
        PORT (
                        wrclk   : IN STD_LOGIC ;
                        rdempty         : OUT STD_LOGIC ;
                        rdreq   : IN STD_LOGIC ;
                        aclr    : IN STD_LOGIC ;
                        wrfull  : OUT STD_LOGIC ;
                        rdclk   : IN STD_LOGIC ;
                        q       : OUT STD_LOGIC_VECTOR (15 DOWNTO
0);
                        wrreq   : IN STD_LOGIC ;
                        data    : IN STD_LOGIC_VECTOR (15 DOWNTO 0);
                        rdusedw         : OUT STD_LOGIC_VECTOR (9
DOWNTO 0)
        );
        END COMPONENT;

    BEGIN
        rdempty    <= sub_wire0;
        wrfull    <= sub_wire1;
```

```
        q    <= sub_wire2(15 DOWNTO 0);
        rdusedw    <= sub_wire3(9 DOWNTO 0);

        dcfifo_component : dcfifo
        GENERIC MAP (
                add_ram_output_register => "OFF",
                clocks_are_synchronized => "FALSE",
                intended_device_family => "Cyclone",
                lpm_hint => "RAM_BLOCK_TYPE=M4K",
                lpm_numwords => 1024, -- Number of words stored in
memory, which is a power of 2.
                lpm_showahead => "ON",
                lpm_type => "dcfifo",
                lpm_width => 16, -- Width of data[] and q[] ports.
                lpm_widthu => 10, -- Width of rdusedw[] and wrusedw[] ports.
                overflow_checking => "ON",
                underflow_checking => "ON",
                use_eab => "ON"
        )
        PORT MAP (
                wrclk => wrclk,
                rdreq => rdreq,
                aclr => aclr,
                rdclk => rdclk,
                wrreq => wrreq,
                data => data,
                rdempty => sub_wire0,
                wrfull => sub_wire1,
                q => sub_wire2,
                rdusedw => sub_wire3
        );



END SYN;
```

GEMRxEventSizeFIFO.vhd

```
-- megafunction wizard: %FIFO%
-- GENERATION: STANDARD
-- VERSION: WM1.0
-- MODULE: dcfifo
```

```
-- ============================================================
========
-- File Name: GEMRxEventSizeFIFO.vhd
-- Megafunction Name(s):
--                      dcfifo
--
-- Simulation Library Files(s):
--                      altera_mf
-- ============================================================
========
-- ************************************************************
*
-- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
--
-- 7.2 Build 207 03/18/2008 SP 3 SJ Full Version
-- ************************************************************
*


--Copyright (C) 1991-2007 Altera Corporation
--Your use of Altera Corporation's design tools, logic functions
--and other software and tools, and its AMPP partner logic
--functions, and any output files from any of the foregoing
--(including device programming or simulation files), and any
--associated documentation or information are expressly subject
--to the terms and conditions of the Altera Program License
--Subscription Agreement, Altera MegaCore Function License
--Agreement, or other applicable license agreement, including,
--without limitation, that your use is for the sole purpose of
--programming logic devices manufactured by Altera and sold by
--Altera or its authorized distributors.  Please refer to the
--applicable agreement for further details.


LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY altera_mf;
USE altera_mf.all;

ENTITY GEMRxEventSizeFIFO IS
        PORT
        (
```

```
		aclr		: IN STD_LOGIC  := '0';
		data		: IN STD_LOGIC_VECTOR (3 DOWNTO 0);
		rdclk		: IN STD_LOGIC ;
		rdreq		: IN STD_LOGIC ;
		wrclk		: IN STD_LOGIC ;
		wrreq		: IN STD_LOGIC ;
		q		: OUT STD_LOGIC_VECTOR (3 DOWNTO
0);
		rdempty			: OUT STD_LOGIC ;
		rdusedw			: OUT STD_LOGIC_VECTOR (5
DOWNTO 0);
		wrfull		: OUT STD_LOGIC
	);
  END GEMRxEventSizeFIFO;


  ARCHITECTURE SYN OF gemrxeventsizefifo IS

	SIGNAL sub_wire0	: STD_LOGIC ;
	SIGNAL sub_wire1	: STD_LOGIC ;
	SIGNAL sub_wire2	: STD_LOGIC_VECTOR (3 DOWNTO 0);
	SIGNAL sub_wire3	: STD_LOGIC_VECTOR (5 DOWNTO 0);



	COMPONENT dcfifo
	GENERIC (
		add_ram_output_register		: STRING;
		clocks_are_synchronized		: STRING;
		intended_device_family		: STRING;
		lpm_numwords		: NATURAL;
		lpm_showahead		: STRING;
		lpm_type		: STRING;
		lpm_width		: NATURAL;
		lpm_widthu		: NATURAL;
		overflow_checking		: STRING;
		underflow_checking		: STRING;
		use_eab		: STRING
	);
	PORT (
			wrclk	: IN STD_LOGIC ;
			rdempty	: OUT STD_LOGIC ;
			rdreq	: IN STD_LOGIC ;
```

```
                        aclr      : IN STD_LOGIC ;
                        wrfull    : OUT STD_LOGIC ;
                        rdclk     : IN STD_LOGIC ;
                        q         : OUT STD_LOGIC_VECTOR (3 DOWNTO
0);
                        wrreq     : IN STD_LOGIC ;
                        data      : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
                        rdusedw          : OUT STD_LOGIC_VECTOR (5
DOWNTO 0)
        );
        END COMPONENT;

    BEGIN
        rdempty    <= sub_wire0;
        wrfull    <= sub_wire1;
        q    <= sub_wire2(3 DOWNTO 0);
        rdusedw    <= sub_wire3(5 DOWNTO 0);

        dcfifo_component : dcfifo
        GENERIC MAP (
                add_ram_output_register => “OFF”,
                clocks_are_synchronized => “FALSE”,
                intended_device_family => “Cyclone”,
                lpm_numwords => 64,
                lpm_showahead => “ON”,
                lpm_type => “dcfifo”,
                lpm_width => 4,
                lpm_widthu => 6,
                overflow_checking => “ON”,
                underflow_checking => “ON”,
                use_eab => “ON”
        )
        PORT MAP (
                wrclk => wrclk,
                rdreq => rdreq,
                aclr => aclr,
                rdclk => rdclk,
                wrreq => wrreq,
                data => data,
                rdempty => sub_wire0,
                wrfull => sub_wire1,
                q => sub_wire2,
                rdusedw => sub_wire3
```

```
        );


    END SYN;
```

GEMRxChannel.vhd

```
    library ieee;
    use ieee.std_logic_1164.all;
    use ieee.std_logic_arith.all;
    use ieee.std_logic_unsigned.all;
    use ieee.std_logic_misc.all;  -- Use OR_REDUCE function

    use work.v1495pkg.all;

    entity GEMRxChannel is
        port(    -- LCLK Synchronous Section
                        LCLK                    : in std_logic;
                        RESET                   : in std_logic;

                        RD_EVENT_SIZE   : in std_logic;
                        EVENT_SIZE              : out std_logic_vector(3 downto
0);
                        EVENT_COUNT             : out std_logic_vector(5
downto 0);

                        RD_EVENT_DATA   : in std_logic;
                        EVENT_DATA              : out std_logic_vector(15
downto 0);
                        EVENT_DATA_SIZE: out std_logic_vector(9 downto
0);

                        GEM_EVENTS_SENT         : out std_logic_vector(31
downto 0);

                        -- CLK Synchronous Section
                        CLK                     : in std_logic;

                        DATA                    : in std_logic;
                        DATA_VALID              : in std_logic
                );
    end GEMRxChannel;
```

```
architecture Synthesis of GEMRxChannel is
	component GEMRxEventSizeFIFO is
		port(
			aclr		: in std_logic := '0';
			data		: in std_logic_vector(3 downto 0);
			rdclk		: in std_logic;
			rdreq		: in std_logic;
			wrclk		: in std_logic;
			wrreq		: in std_logic;
			q			: out std_logic_vector(3 downto
0);
			rdempty		: out std_logic;
			rdusedw		: out std_logic_vector(5 downto
0);
			wrfull		: out std_logic
		);
	end component;

	component GEMRxEventDataFIFO is
		port(
			aclr		: in std_logic := '0';
			data		: in std_logic_vector(15 downto 0);
			rdclk		: in std_logic;
			rdreq		: in std_logic;
			wrclk		: in std_logic;
			wrreq		: in std_logic;
			q			: out std_logic_vector (15 downto
0);
			rdempty		: out std_logic;
			rdusedw		: out std_logic_vector (9 downto
0);
			wrfull		: out std_logic
		);
	end component;

	constant WORD_COUNT_MAX		: std_logic_vector(3
downto 0) := "1100";

	signal SHIFT_REG		: std_logic_vector(15 downto 0);

	signal CAPTURE_BITS		: std_logic;
```

```
        signal DATA_VALID_UP            : std_logic;
        signal DATA_VALID_DOWN                  : std_logic;
        signal DATA_VALID_Q                     : std_logic;

        signal BIT_COUNTER                      : std_logic_vector(3
downto 0);
        signal BIT_COUNTER_OVF          : std_logic;
        signal BIT_COUNTER_OVF_Q        : std_logic;

        signal WORD_COUNTER                     : std_logic_vector(3
downto 0);
        signal WORD_INHIBIT                     : std_logic;

        signal EVENT_SIZE_FIFO_FULL   : std_logic;
        signal EVENT_DATA_FIFO_FULL : std_logic;

        signal EVENT_SIZE_FIFO_WR       : std_logic;

        signal GEM_EVENTS_SENT_i                : std_logic_vector(31
downto 0);
        signal DATA_VALID_LCLK_Q1       : std_logic;
        signal DATA_VALID_LCLK_Q2       : std_logic;
   begin

        GEM_EVENTS_SENT <= GEM_EVENTS_SENT_i;

        process(RESET, LCLK)
        begin
                if RESET = '1' then
                        GEM_EVENTS_SENT_i <= (others => '0');
                        DATA_VALID_LCLK_Q1 <= '0';
                        DATA_VALID_LCLK_Q2 <= '0';
                elsif rising_edge(LCLK) then
                        DATA_VALID_LCLK_Q1 <= DATA_VALID;
                        DATA_VALID_LCLK_Q2 <= DATA_VALID_LCLK_
Q1;
                        if DATA_VALID_LCLK_Q1 = '1' and DATA_VALID_
LCLK_Q2 = '0' then
                                GEM_EVENTS_SENT_i <= GEM_EVENTS_
SENT_i + 1;
                        end if;
                end if;
        end process;
```

```
        DATA_VALID_UP <= DATA_VALID and not DATA_VALID_Q;
        DATA_VALID_DOWN <= DATA_VALID_Q and not DATA_VALID;

        EVENT_SIZE_FIFO_WR <= DATA_VALID_DOWN and
CAPTURE_BITS;

        process(CLK)
        begin
                if rising_edge(CLK) then
                        if RESET = '1' or DATA_VALID_DOWN = '1' then
                                CAPTURE_BITS <= '0';
                        elsif DATA_VALID_UP = '1' and EVENT_SIZE_
FIFO_FULL = '0' and EVENT_DATA_FIFO_FULL = '0' then
                                CAPTURE_BITS <= '1';
                        end if;
                end if;
        end process;

        BIT_COUNTER_OVF <= '1' when BIT_COUNTER = "1111" and
WORD_INHIBIT = '0' and CAPTURE_BITS = '1' else '0';

        process(CLK)
        begin
                if rising_edge(CLK) then
                        DATA_VALID_Q <= DATA_VALID;
                        BIT_COUNTER_OVF_Q <= BIT_COUNTER_OVF;
                end if;
        end process;

        process(CLK)
        begin
                if rising_edge(CLK) then
                        if RESET = '1' or DATA_VALID = '0' then
                                BIT_COUNTER <= "0000";
                        else
                                BIT_COUNTER <= BIT_COUNTER + 1;
                        end if;
                end if;
        end process;

        WORD_INHIBIT <= '1' when WORD_COUNTER = WORD_
COUNT_MAX else '0';
```

```
	process(CLK)
	begin
		if rising_edge(CLK) then
			if RESET = '1' or DATA_VALID_UP = '1' then
				WORD_COUNTER <= "0000";
			elsif BIT_COUNTER_OVF = '1' then
				WORD_COUNTER <= WORD_COUNTER +
1;
			end if;
		end if;
	end process;

	process(CLK)
	begin
		if rising_edge(CLK) then
			if DATA_VALID = '1' then
				SHIFT_REG <= SHIFT_REG(14 downto 0) &
DATA;
			end if;
		end if;
	end process;

	GEMRxEventSizeFIFO_0: GEMRxEventSizeFIFO
		port map(	aclr	=> RESET,
					data	=> WORD_COUNTER,
					rdclk	=> LCLK,
					rdreq	=> RD_EVENT_SIZE,
					wrclk	=> CLK,
					wrreq	=> EVENT_SIZE_FIFO_WR,
					q		=> EVENT_SIZE,
					rdempty	=> open,
					rdusedw	=> EVENT_COUNT,
					wrfull	=> EVENT_SIZE_FIFO_FULL
		);

	GEMRxEventDataFIFO_0: GEMRxEventDataFIFO
		port map(	aclr	=> RESET,
					data	=> SHIFT_REG,  -- Data input to
the dcfifo.
                                                                --
(Input port LPM_WIDTH wide.)
					rdclk	=> LCLK,
```

```
                                        rdreq   => RD_EVENT_DATA, -- Read
request. The oldest data in the dcfifo

  --   goes to the q[] port.

  -- (Reading is disabled if the value of

  --  the rdempty output port is 1.)


                                        wrclk   => CLK,

                                        wrreq   => BIT_COUNTER_OVF_Q, --
Write request. The data[] port is written to

      --  the dcfifo.

      -- (Writing is disabled if wrfull = 1.)
                                        q               => EVENT_DATA, --
Data output from the dcfifo.
                                                                        --
(Output port LPM_WIDTH wide.)
                                        rdempty         => open,
                                        rdusedw         => EVENT_DATA_SIZE,
-- Number of words that are currently in FIFO

            --          (Synchronized to rdclk)
                                        wrfull  => EVENT_DATA_FIFO_FULL
-- If asserted, indicates that the dcfifo is

                  --  full and disables the wrreq port.

                  -- (Synchronized with wrclk.)

                  );

    end Synthesis;
```

GEMTxChannel.vhd

```
    library ieee;
    use ieee.std_logic_1164.all;
    use ieee.std_logic_arith.all;
```

```
use ieee.std_logic_unsigned.all;
use ieee.std_logic_misc.all;  -- Use OR_REDUCE function

use work.v1495pkg.all;

entity GEMTxChannel is
        port(    -- LCLK Synchronous Section
                        LCLK                    : in std_logic;

        GEM_TX_WORD_0 : in std_logic_vector(15 downto 0);
        GEM_TX_WORD_1 : in std_logic_vector(15 downto 0);
        GEM_TX_WORD_2 : in std_logic_vector(15 downto 0);
        GEM_TX_WORD_3 : in std_logic_vector(15 downto 0);
        GEM_TX_WORD_4 : in std_logic_vector(15 downto 0);
        GEM_TX_WORD_5 : in std_logic_vector(15 downto 0);
        GEM_TX_WORD_6 : in std_logic_vector(15 downto 0);
        GEM_TX_WORD_7 : in std_logic_vector(15 downto 0);
        GEM_TX_WORD_8 : in std_logic_vector(15 downto 0);
        GEM_TX_WORD_9 : in std_logic_vector(15 downto 0);
        GEM_TX_WORD_A : in std_logic_vector(15 downto 0);
        GEM_TX_WORD_B : in std_logic_vector(15 downto 0);

        GEM_TX_START     : in std_logic;

        -- CLK Synchronous Section
        CLK: in std_logic;

        TX_EXT_EN: in std_logic;
        HARD_TRIGGER     : in std_logic;

        DATA  : out std_logic;
        DATA_VALID: out std_logic
);
end GEMTxChannel;

architecture Synthesis of GEMTxChannel is
        constant GEM_TX_FRAME_LEN    : std_logic_vector(7 downto 0) :=
"11100000";

        signal GEM_TX_SHIFT_REG                 : std_logic_vector(191
downto 0) := x"000000000000000000000000000000000000000000000000";

        signal BIT_COUNTER                      : std_logic_vector(7
```

```
downto 0) := “00000000”;
        signal BIT_COUNTER_DONE                 : std_logic;

        signal GEM_TX_START_Q1          : std_logic;
        signal GEM_TX_START_Q2          : std_logic;
        signal GEM_TX_START_UP          : std_logic;

        signal HARD_TRIGGER_Q1                  : std_logic;
        signal EXT_TX_START                     : std_logic;
   begin

        EXT_TX_START <= TX_EXT_EN and HARD_TRIGGER and not
HARD_TRIGGER_Q1;
        GEM_TX_START_UP <= GEM_TX_START_Q1 and not GEM_TX_
START_Q2;

        process(CLK)
        begin
                if rising_edge(CLK) then
                        HARD_TRIGGER_Q1 <= HARD_TRIGGER;
                        GEM_TX_START_Q1 <= GEM_TX_START;
                        GEM_TX_START_Q2 <= GEM_TX_START_Q1;
                end if;
        end process;

        process(CLK)
        begin
                if rising_edge(CLK) then
                        if BIT_COUNTER_DONE = ‘1’ and (GEM_TX_
START_UP = ‘1’ or EXT_TX_START = ‘1’) then
                                GEM_TX_SHIFT_REG <= GEM_TX_
WORD_0 & GEM_TX_WORD_1 & GEM_TX_WORD_2 & GEM_TX_
WORD_3 &
                                        GEM_TX_WORD_4 & GEM_TX_
WORD_5 & GEM_TX_WORD_6 & GEM_TX_WORD_7 &
                                        GEM_TX_WORD_8 & GEM_TX_
WORD_9 & GEM_TX_WORD_A & GEM_TX_WORD_B;
                        else
                                GEM_TX_SHIFT_REG <= GEM_TX_SHIFT_
REG(190 downto 0) & ‘0’;
                        end if;
                end if;
        end process;
```

```
        BIT_COUNTER_DONE <= '1' when BIT_COUNTER = "00000000"
else '0';

        process(CLK)
        begin
                if rising_edge(CLK) then
                        if BIT_COUNTER_DONE = '1' and (GEM_TX_
START_UP = '1' or EXT_TX_START = '1') then
                                BIT_COUNTER <= GEM_TX_FRAME_LEN;
                        elsif BIT_COUNTER_DONE = '0' then
                                BIT_COUNTER <= BIT_COUNTER - 1;
                        end if;
                end if;
        end process;

        process(CLK)
        begin
                if rising_edge(CLK) then
                        DATA <= GEM_TX_SHIFT_REG(191);
                        DATA_VALID <= not BIT_COUNTER_DONE;
                end if;
        end process;

   end Synthesis;
```

GEMReadout_tb.vhd

```
   library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_misc.all;  -- Use OR_REDUCE function

use work.v1495pkg.all;

entity GEMReadout_tb is
end GEMReadout_tb;

architecture TestBench of GEMReadout_tb is

        component GEMReadout is
                port(
```

```
        nLBRES  : IN   std_logic ;       -- Async Reset (active low)
        LCLK    : IN   std_logic ;       -- Local Bus Clock
        REG_WREN  : IN   std_logic ;       -- Write pulse (active high)
        REG_RDEN  : IN   std_logic ;       -- Read  pulse (active high)
        REG_ADDR  : IN   std_logic_vector (15 DOWNTO 0); -- Register
address
        REG_DIN   : IN   std_logic_vector (15 DOWNTO 0); -- Data from
CAEN Local Bus
        REG_DOUT  : OUT  std_logic_vector (15 DOWNTO 0); -- Data TO
CAEN Local Bus
        USR_ACCESS  : IN   std_logic ;       -- Current register access is
        A     : IN   std_logic_vector (31 DOWNTO 0); -- In A (32 x LVDS/
ECL)
        B     : IN   std_logic_vector (31 DOWNTO 0); -- In B (32 x LVDS/
ECL)
        C     : OUT  std_logic_vector (31 DOWNTO 0); -- Out C (32 x LVDS)
        SELG    : OUT  std_logic ;       -- Output Level Select (NIM/TTL)
        nOEG    : OUT  std_logic ;       -- Output Enable
        GOUT    : OUT  std_logic_vector (1 DOWNTO 0);  -- Out G - LEMO
(2 x NIM/TTL)
        GIN   : IN   std_logic_vector (1 DOWNTO 0);  -- In G - LEMO (2 x
NIM/TTL)
        IDD   : IN   std_logic_vector ( 2 DOWNTO 0); -- D slot mezzanine
Identifier
        SELD    : OUT  std_logic ;       -- D slot Port Signal Standard Select
        nOED    : OUT  std_logic ;       -- D slot Port Direction
        D     : IN   std_logic_vector (31 DOWNTO 0); -- D slot Data In  Bus
        IDE   : IN   std_logic_vector ( 2 DOWNTO 0); -- E slot mezzanine
Identifier
        SELE    : OUT  std_logic ;       -- E slot Port Signal Standard Select
        nOEE    : OUT  std_logic ;       -- E slot Port Direction
        E     : OUT  std_logic_vector (31 DOWNTO 0); -- E slot Data In  Bus
        IDF   : IN   std_logic_vector ( 2 DOWNTO 0); -- F slot mezzanine
Identifier
        SELF    : OUT  std_logic ;       -- F slot Port Signal Standard Select
        nOEF    : OUT  std_logic ;       -- F slot Port Direction
        F     : IN   std_logic_vector (31 DOWNTO 0); -- F slot Data Out Bus
        PDL_WR  : OUT  std_logic ;       -- Write Enable
        PDL_SEL   : OUT  std_logic ;       -- PDL Selection (0=>PDL0,
1=>PDL1)
        PDL_READ  : IN   std_logic_vector ( 7 DOWNTO 0); -- Read Data
        PDL_WRITE : OUT  std_logic_vector ( 7 DOWNTO 0); -- Write Data
        PDL_DIR   : OUT  std_logic ;       -- Direction (0=>Write, 1=>Read)
```

```
        PDL0_OUT  : IN   std_logic ;       -- Signal from PDL0 Output
        PDL1_OUT  : IN   std_logic ;       -- Signal from PDL1 Output
        DLO0_OUT  : IN   std_logic ;       -- Signal from DLO0 Output
        DLO1_OUT  : IN   std_logic ;       -- Signal from DLO1 Output
        PDL0_IN   : OUT  std_logic ;       -- Signal TO PDL0 Input
        PDL1_IN   : OUT  std_logic ;       -- Signal TO PDL1 Input
        DLO0_GATE : OUT  std_logic ;       -- DLO0 Gate (active high)
        DLO1_GATE : OUT  std_logic ;       -- DLO1 Gate (active high)
        SPARE_OUT : OUT  std_logic_vector (11 DOWNTO 0); -- SPARE
Data Out
        SPARE_IN  : IN   std_logic_vector (11 DOWNTO 0); -- SPARE Data
In
        SPARE_DIR : OUT  std_logic_vector (11 DOWNTO 0); -- SPARE
Direction (0 => OUT, 1 => IN)
        RED_PULSE : OUT  std_logic ;       -- RED Led Pulse (active high)
        GREEN_PULSE : OUT  std_logic        -- GREEN Led Pulse (active
high)
);
        end component;

        signal nLBRES      : std_logic;
        signal LCLK        : std_logic;
        signal REG_WREN    : std_logic;
        signal REG_RDEN    : std_logic;
        signal REG_ADDR    : std_logic_vector(15 DOWNTO 0);
        signal REG_DIN     : std_logic_vector(15 DOWNTO 0);
        signal REG_DOUT    : std_logic_vector(15 DOWNTO 0);
        signal USR_ACCESS  : std_logic;
        signal A           : std_logic_vector(31 DOWNTO 0);
        signal B           : std_logic_vector(31 DOWNTO 0);
        signal C           : std_logic_vector(31 DOWNTO 0);
        signal SELG        : std_logic;
        signal nOEG        : std_logic;
        signal GOUT        : std_logic_vector(1 DOWNTO 0);
        signal GIN         : std_logic_vector(1 DOWNTO 0);
        signal IDD         : std_logic_vector(2 DOWNTO 0);
        signal SELD        : std_logic;
        signal nOED        : std_logic;
        signal D           : std_logic_vector(31 DOWNTO 0);
        signal IDE         : std_logic_vector(2 DOWNTO 0);
        signal SELE        : std_logic;
        signal nOEE        : std_logic;
        signal E           : std_logic_vector(31 DOWNTO 0);
```

```
        signal IDF        : std_logic_vector(2 DOWNTO 0);
        signal SELF       : std_logic;
        signal nOEF       : std_logic;
        signal F          : std_logic_vector(31 DOWNTO 0);
        signal PDL_WR     : std_logic;
        signal PDL_SEL    : std_logic;
        signal PDL_READ   : std_logic_vector(7 DOWNTO 0);
        signal PDL_WRITE  : std_logic_vector(7 DOWNTO 0);
        signal PDL_DIR    : std_logic;
        signal PDL0_OUT   : std_logic;
        signal PDL1_OUT   : std_logic;
        signal DLO0_OUT   : std_logic;
        signal DLO1_OUT   : std_logic;
        signal PDL0_IN    : std_logic;
        signal PDL1_IN    : std_logic;
        signal DLO0_GATE  : std_logic;
        signal DLO1_GATE  : std_logic;
        signal SPARE_OUT  : std_logic_vector(11 DOWNTO 0);
        signal SPARE_IN   : std_logic_vector(11 DOWNTO 0);
        signal SPARE_DIR  : std_logic_vector(11 DOWNTO 0);
        signal RED_PULSE  : std_logic;
        signal GREEN_PULSE : std_logic;

        constant NUM_EVENTS             : integer := 64;          -- Internal
FIFO's can hold 63 events
begin

        GEMReadout_UUT: GEMReadout
                port map(
                        nLBRES              => nLBRES,
                        LCLK          => LCLK,
                        REG_WREN      => REG_WREN,
                        REG_RDEN      => REG_RDEN,
                        REG_ADDR      => REG_ADDR,
                        REG_DIN             => REG_DIN,
                        REG_DOUT      => REG_DOUT,
                        USR_ACCESS          => USR_ACCESS,
                        A                   => A,
                        B                   => B,
                        C                   => C,
                        SELG          => SELG,
                        nOEG          => nOEG,
                        GOUT          => GOUT,
```

```
                GIN                     => GIN,
                IDD                     => IDD,
                SELD            => SELD,
                nOED            => nOED,
                D                       => D,
                IDE                     => IDE,
                SELE            => SELE,
                nOEE            => nOEE,
                E                       => E,
                IDF                     => IDF,
                SELF            => SELF,
                nOEF            => nOEF,
                F                       => F,
                PDL_WR                  => PDL_WR,
                PDL_SEL                 => PDL_SEL,
                PDL_READ        => PDL_READ,
                PDL_WRITE       => PDL_WRITE,
                PDL_DIR                 => PDL_DIR,
                PDL0_OUT        => PDL0_OUT,
                PDL1_OUT        => PDL1_OUT,
                DLO0_OUT        => DLO0_OUT,
                DLO1_OUT        => DLO1_OUT,
                PDL0_IN                 => PDL0_IN,
                PDL1_IN                 => PDL1_IN,
                DLO0_GATE       => DLO0_GATE,
                DLO1_GATE       => DLO1_GATE,
                SPARE_OUT       => SPARE_OUT,
                SPARE_IN        => SPARE_IN,
                SPARE_DIR       => SPARE_DIR,
                RED_PULSE       => RED_PULSE,
                GREEN_PULSE             => GREEN_PULSE
        );

USR_ACCESS <= '1';
B <= (others => '0');
D <= (others => '0');
F <= (others => '0');
PDL_READ <= (others => '0');
PDL0_OUT <= '0';
PDL1_OUT <= '0';
DLO0_OUT <= '0';
DLO1_OUT <= '0';
SPARE_IN <= (others => '0');
```

```
	process
	begin
		LCLK <= '1';
		wait for 12.5 ns;
		LCLK <= '0';
		wait for 12.5 ns;
	end process;

	process
	begin
		nLBRES <= '0';
		wait for 100 ns;
		nLBRES <= '1';
		wait;
	end process;

	A <= transport E after 16 ns;
--	C

	process
	begin
		GIN(0) <= '1';
		wait for 15.625 ns;
		GIN(0) <= '0';
		wait for 15.625 ns;
	end process;

	process
	begin
		GIN(1) <= '0';
		wait for 1 us;
		GIN(1) <= '1';
		wait for 50 ns;
	end process;

	process
		procedure UWrite(ADDR : in std_logic_vector(15 downto 0);
DATA : in std_logic_vector(15 downto 0)) is
		begin
			wait until falling_edge(LCLK);
			REG_WREN <= '1';
			REG_ADDR <= ADDR;
```

```
                REG_DIN <= DATA;
                wait until falling_edge(LCLK);
                REG_WREN <= '0';
                wait for 10 ns;
        end procedure;

        procedure URead(ADDR : in std_logic_vector(15 downto 0)) is
        begin
                wait until falling_edge(LCLK);
                REG_RDEN <= '1';
                REG_ADDR <= ADDR;
                wait until falling_edge(LCLK);
                REG_RDEN <= '0';
                wait for 10 ns;
        end procedure;

        procedure UReadEvent(ADDR : in std_logic_vector(15 downto
0)) is
        begin
                for I in 1 to 12 loop
                        URead(ADDR);
                end loop;
        end procedure;
    begin
        REG_WREN <= '0';
        REG_RDEN <= '0';
        REG_ADDR <= (others => '0');
        REG_DIN <= (others => '0');
        wait for 1 us;

        UWrite(A_GEM_TRIG_WORD, "0000000000" & "100" &
"110");-- Trigger Word Definitions
        wait for 200 ns;
        UWrite(A_GEM_SOFT_TRIG, x"0000");
        -- Soft Trigger
        wait for 200 ns;

        -- GEM Tx Test Frame Definition
        UWrite(A_GEM_TX_WORD_0, "1010" & x"012");
        UWrite(A_GEM_TX_WORD_1, "1100" & x"345");
        UWrite(A_GEM_TX_WORD_2, "1110" & x"678");
        UWrite(A_GEM_TX_WORD_3, x"9ABC");
        UWrite(A_GEM_TX_WORD_4, x"DEF0");
```

```
UWrite(A_GEM_TX_WORD_5, x”1234”);
UWrite(A_GEM_TX_WORD_6, x”5678”);
UWrite(A_GEM_TX_WORD_7, x”9ABC”);
UWrite(A_GEM_TX_WORD_8, x”DEF0”);
UWrite(A_GEM_TX_WORD_9, x”1234”);
UWrite(A_GEM_TX_WORD_A, x”5678”);
UWrite(A_GEM_TX_WORD_B, x”9ABC”);
wait for 1 us;

-- Check Event Counters (should be 0 here)
URead(A_GEMA0_FIFOSIZE);
URead(A_GEMA1_FIFOSIZE);
URead(A_GEMA2_FIFOSIZE);
URead(A_GEMA3_FIFOSIZE);
URead(A_GEMA4_FIFOSIZE);
URead(A_GEMA5_FIFOSIZE);
URead(A_GEMB0_FIFOSIZE);
URead(A_GEMB1_FIFOSIZE);
URead(A_GEMB2_FIFOSIZE);
URead(A_GEMB3_FIFOSIZE);
URead(A_GEMB4_FIFOSIZE);
URead(A_GEMB5_FIFOSIZE);

-- Generate GEM Tx Test Frame
for I in 1 to NUM_EVENTS loop
        UWrite(A_GEM_TX_START, x”0000”);
        wait for 6 us;
end loop;

-- Check Event Counters (should be NUM_EVENTS here)
URead(A_GEMA0_FIFOSIZE);
URead(A_GEMA1_FIFOSIZE);
URead(A_GEMA2_FIFOSIZE);
URead(A_GEMA3_FIFOSIZE);
URead(A_GEMA4_FIFOSIZE);
URead(A_GEMA5_FIFOSIZE);
URead(A_GEMB0_FIFOSIZE);
URead(A_GEMB1_FIFOSIZE);
URead(A_GEMB2_FIFOSIZE);
URead(A_GEMB3_FIFOSIZE);
URead(A_GEMB4_FIFOSIZE);
URead(A_GEMB5_FIFOSIZE);
```

```
-- Readout Events
for I in 1 to NUM_EVENTS loop
        -- Get Event Size
        URead(A_GEMA0_EVENTSIZE);
        URead(A_GEMA1_EVENTSIZE);
        URead(A_GEMA2_EVENTSIZE);
        URead(A_GEMA3_EVENTSIZE);
        URead(A_GEMA4_EVENTSIZE);
        URead(A_GEMA5_EVENTSIZE);
        URead(A_GEMB0_EVENTSIZE);
        URead(A_GEMB1_EVENTSIZE);
        URead(A_GEMB2_EVENTSIZE);
        URead(A_GEMB3_EVENTSIZE);
        URead(A_GEMB4_EVENTSIZE);
        URead(A_GEMB5_EVENTSIZE);

        UReadEvent(A_GEMA0_EVENTDATA);
        UReadEvent(A_GEMA1_EVENTDATA);
        UReadEvent(A_GEMA2_EVENTDATA);
        UReadEvent(A_GEMA3_EVENTDATA);
        UReadEvent(A_GEMA4_EVENTDATA);
        UReadEvent(A_GEMA5_EVENTDATA);
        UReadEvent(A_GEMB0_EVENTDATA);
        UReadEvent(A_GEMB1_EVENTDATA);
        UReadEvent(A_GEMB2_EVENTDATA);
        UReadEvent(A_GEMB3_EVENTDATA);
        UReadEvent(A_GEMB4_EVENTDATA);
        UReadEvent(A_GEMB5_EVENTDATA);
end loop;

-- Check Event Counters (should be 0 here)
URead(A_GEMA0_FIFOSIZE);
URead(A_GEMA1_FIFOSIZE);
URead(A_GEMA2_FIFOSIZE);
URead(A_GEMA3_FIFOSIZE);
URead(A_GEMA4_FIFOSIZE);
URead(A_GEMA5_FIFOSIZE);
URead(A_GEMB0_FIFOSIZE);
URead(A_GEMB1_FIFOSIZE);
URead(A_GEMB2_FIFOSIZE);
URead(A_GEMB3_FIFOSIZE);
URead(A_GEMB4_FIFOSIZE);
URead(A_GEMB5_FIFOSIZE);
```

```
            wait;
        end process;

end TestBench;
```

v1495usr_hal.vqm

```
// 
// Written by Synplify
// Synplify 8.1.0, Build 539R.
// Wed Aug 02 15:09:51 2006
//
// Source file index table:
// Object locations will have the form <file>:<line>
// file 0 “noname”
// file 1 “\c:\programmi\synplicity\fpga_81\lib\vhd\std.vhd “
// file 2 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\v1495usr_pkg.
vhd “
// file 3 “\c:\programmi\synplicity\fpga_81\lib\vhd\std1164.vhd “
// file 4 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\fpga_if_rtl.vhd “
// file 5 “\c:\programmi\synplicity\fpga_81\lib\vhd\arith.vhd “
// file 6 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\a395x_if_rtl.vhd
“
// file 7 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\coin_reference.
vhd “
// file 8 “\c:\programmi\synplicity\fpga_81\lib\vhd\unsigned.vhd “
// file 9 “\c:\programmi\synplicity\fpga_81\lib\vhd\misc.vhd “
// file 10 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\tristate_if_rtl.
vhd “
// file 11 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\delay_if_rtl.vhd
“
// file 12 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\frontpan_if_rtl.
vhd “
// file 13 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\led_if_rtl.vhd “
// file 14 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\localbusif.vhd “
// file 15 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\pdl_if.vhd “
// file 16 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\spare_if_rtl.vhd
“
// file 17 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\v1495usr_hal.
vhd “
// file 18 “\c:\luca\work\v1495\fpga\src\v1495usr_demo\hdl\v1495usr_
demo.vhd “
```

```
// VQM4.1+
module led_if (
  TCNT2_7,
  TCNT2_6,
  TCNT2_5,
  TCNT2_4,
  TCNT2_3,
  TCNT2_2,
  TCNT2_1,
  TCNT2_0,
  TCNT1_2,
  TCNT1_1,
  TCNT1_0,
  nLRESET_i_x_i,
  nLEDG,
  nLEDR,
  GREEN_PULSE,
  RED_PULSE,
  nLRESET,
  un7_tcnt1_a,
  un14_tcnt3_5_5,
  un14_tcnt3_5_4,
  LCLK
);
output TCNT2_7 ;
output TCNT2_6 ;
output TCNT2_5 ;
output TCNT2_4 ;
output TCNT2_3 ;
output TCNT2_2 ;
output TCNT2_1 ;
output TCNT2_0 ;
output TCNT1_2 ;
output TCNT1_1 ;
output TCNT1_0 ;
input nLRESET_i_x_i ;
output nLEDG ;
output nLEDR ;
input GREEN_PULSE ;
input RED_PULSE ;
input nLRESET ;
input un7_tcnt1_a ;
```

```
input un14_tcnt3_5_5 ;
input un14_tcnt3_5_4 ;
input LCLK ;
wire TCNT2_7 ;
wire TCNT2_6 ;
wire TCNT2_5 ;
wire TCNT2_4 ;
wire TCNT2_3 ;
wire TCNT2_2 ;
wire TCNT2_1 ;
wire TCNT2_0 ;
wire TCNT1_2 ;
wire TCNT1_1 ;
wire TCNT1_0 ;
wire nLRESET_i_x_i ;
wire nLEDG ;
wire nLEDR ;
wire GREEN_PULSE ;
wire RED_PULSE ;
wire nLRESET ;
wire un7_tcnt1_a ;
wire un14_tcnt3_5_5 ;
wire un14_tcnt3_5_4 ;
wire LCLK ;
wire [4:0] TCNT1_cout;
wire [5:3] TCNT1;
wire [6:0] TCNT2_cout;
wire [2:0] TICK;
wire [7:0] TCNT3;
wire [6:0] TCNT3_cout;
wire VCC ;
wire un14_tcnt3_4 ;
wire un14_tcnt3_5_0 ;
wire un14_tcnt3_5 ;
wire un7_tcnt1 ;
wire un2_red_pulse_0_a2_x ;
wire un2_green_pulse_0_a2_x ;
wire nLEDR_0_0 ;
wire TMONOSR ;
wire nLEDG_0_0 ;
wire TMONOSG ;
wire TMONOSR_0_0_a2 ;
wire TMONOSG_0_0_a2 ;
```

```
wire GND ;
wire nLRESET_i ;
//@1:1
  assign VCC = 1'b1;
  assign GND = 1'b0;
// @13:107
  cyclone_lcell TCNT1_0__Z (
        .regout(TCNT1_0),
        .cout(TCNT1_cout[0]),
        .clk(LCLK),
        .dataa(TCNT1_0),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT1_0__Z.operation_mode="arithmetic";
defparam TCNT1_0__Z.output_mode="reg_only";
defparam TCNT1_0__Z.lut_mask="55aa";
defparam TCNT1_0__Z.synch_mode="off";
defparam TCNT1_0__Z.sum_lutc_input="datac";
// @13:107
  cyclone_lcell TCNT1_1__Z (
        .regout(TCNT1_1),
        .cout(TCNT1_cout[1]),
        .clk(LCLK),
        .dataa(TCNT1_1),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .cin(TCNT1_cout[0]),
        .inverta(GND),
        .aload(GND),
```

```
        .regcascin(GND)
);
defparam TCNT1_1__Z.cin_used=”true”;
defparam TCNT1_1__Z.operation_mode=”arithmetic”;
defparam TCNT1_1__Z.output_mode=”reg_only”;
defparam TCNT1_1__Z.lut_mask=”5aa0”;
defparam TCNT1_1__Z.synch_mode=”off”;
defparam TCNT1_1__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT1_2__Z (
        .regout(TCNT1_2),
        .cout(TCNT1_cout[2]),
        .clk(LCLK),
        .dataa(TCNT1_2),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .cin(TCNT1_cout[1]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT1_2__Z.cin_used=”true”;
defparam TCNT1_2__Z.operation_mode=”arithmetic”;
defparam TCNT1_2__Z.output_mode=”reg_only”;
defparam TCNT1_2__Z.lut_mask=”5aa0”;
defparam TCNT1_2__Z.synch_mode=”off”;
defparam TCNT1_2__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT1_3__Z (
        .regout(TCNT1[3]),
        .cout(TCNT1_cout[3]),
        .clk(LCLK),
        .dataa(TCNT1[3]),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
```

```
        .sload(GND),
        .ena(VCC),
        .cin(TCNT1_cout[2]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT1_3__Z.cin_used=”true”;
defparam TCNT1_3__Z.operation_mode=”arithmetic”;
defparam TCNT1_3__Z.output_mode=”reg_only”;
defparam TCNT1_3__Z.lut_mask=”5aa0”;
defparam TCNT1_3__Z.synch_mode=”off”;
defparam TCNT1_3__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT1_4__Z (
        .regout(TCNT1[4]),
        .cout(TCNT1_cout[4]),
        .clk(LCLK),
        .dataa(TCNT1[4]),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .cin(TCNT1_cout[3]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT1_4__Z.cin_used=”true”;
defparam TCNT1_4__Z.operation_mode=”arithmetic”;
defparam TCNT1_4__Z.output_mode=”reg_only”;
defparam TCNT1_4__Z.lut_mask=”5aa0”;
defparam TCNT1_4__Z.synch_mode=”off”;
defparam TCNT1_4__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT1_5__Z (
        .regout(TCNT1[5]),
        .clk(LCLK),
        .dataa(TCNT1[5]),
        .datab(VCC),
```

```
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .cin(TCNT1_cout[4]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT1_5__Z.cin_used="true";
defparam TCNT1_5__Z.operation_mode="normal";
defparam TCNT1_5__Z.output_mode="reg_only";
defparam TCNT1_5__Z.lut_mask="5a5a";
defparam TCNT1_5__Z.synch_mode="off";
defparam TCNT1_5__Z.sum_lutc_input="cin";
// @13:107
  cyclone_lcell TCNT2_0__Z (
        .regout(TCNT2_0),
        .cout(TCNT2_cout[0]),
        .clk(LCLK),
        .dataa(TCNT2_0),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[0]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT2_0__Z.operation_mode="arithmetic";
defparam TCNT2_0__Z.output_mode="reg_only";
defparam TCNT2_0__Z.lut_mask="55aa";
defparam TCNT2_0__Z.synch_mode="off";
defparam TCNT2_0__Z.sum_lutc_input="datac";
// @13:107
  cyclone_lcell TCNT2_1__Z (
        .regout(TCNT2_1),
        .cout(TCNT2_cout[1]),
```

```
        .clk(LCLK),
        .dataa(TCNT2_1),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[0]),
        .cin(TCNT2_cout[0]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT2_1__Z.cin_used=”true”;
defparam TCNT2_1__Z.operation_mode=”arithmetic”;
defparam TCNT2_1__Z.output_mode=”reg_only”;
defparam TCNT2_1__Z.lut_mask=”5aa0”;
defparam TCNT2_1__Z.synch_mode=”off”;
defparam TCNT2_1__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT2_2__Z (
        .regout(TCNT2_2),
        .cout(TCNT2_cout[2]),
        .clk(LCLK),
        .dataa(TCNT2_2),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[0]),
        .cin(TCNT2_cout[1]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT2_2__Z.cin_used=”true”;
defparam TCNT2_2__Z.operation_mode=”arithmetic”;
defparam TCNT2_2__Z.output_mode=”reg_only”;
defparam TCNT2_2__Z.lut_mask=”5aa0”;
defparam TCNT2_2__Z.synch_mode=”off”;
```

```
defparam TCNT2_2__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT2_3__Z (
        .regout(TCNT2_3),
        .cout(TCNT2_cout[3]),
        .clk(LCLK),
        .dataa(TCNT2_3),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[0]),
        .cin(TCNT2_cout[2]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT2_3__Z.cin_used=”true”;
defparam TCNT2_3__Z.operation_mode=”arithmetic”;
defparam TCNT2_3__Z.output_mode=”reg_only”;
defparam TCNT2_3__Z.lut_mask=”5aa0”;
defparam TCNT2_3__Z.synch_mode=”off”;
defparam TCNT2_3__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT2_4__Z (
        .regout(TCNT2_4),
        .cout(TCNT2_cout[4]),
        .clk(LCLK),
        .dataa(TCNT2_4),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[0]),
        .cin(TCNT2_cout[3]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
```

```
defparam TCNT2_4__Z.cin_used=”true”;
defparam TCNT2_4__Z.operation_mode=”arithmetic”;
defparam TCNT2_4__Z.output_mode=”reg_only”;
defparam TCNT2_4__Z.lut_mask=”5aa0”;
defparam TCNT2_4__Z.synch_mode=”off”;
defparam TCNT2_4__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT2_5__Z (
        .regout(TCNT2_5),
        .cout(TCNT2_cout[5]),
        .clk(LCLK),
        .dataa(TCNT2_5),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[0]),
        .cin(TCNT2_cout[4]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT2_5__Z.cin_used=”true”;
defparam TCNT2_5__Z.operation_mode=”arithmetic”;
defparam TCNT2_5__Z.output_mode=”reg_only”;
defparam TCNT2_5__Z.lut_mask=”5aa0”;
defparam TCNT2_5__Z.synch_mode=”off”;
defparam TCNT2_5__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT2_6__Z (
        .regout(TCNT2_6),
        .cout(TCNT2_cout[6]),
        .clk(LCLK),
        .dataa(TCNT2_6),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[0]),
```

```
        .cin(TCNT2_cout[5]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT2_6__Z.cin_used="true";
defparam TCNT2_6__Z.operation_mode="arithmetic";
defparam TCNT2_6__Z.output_mode="reg_only";
defparam TCNT2_6__Z.lut_mask="5aa0";
defparam TCNT2_6__Z.synch_mode="off";
defparam TCNT2_6__Z.sum_lutc_input="cin";
// @13:107
  cyclone_lcell TCNT2_7__Z (
        .regout(TCNT2_7),
        .clk(LCLK),
        .dataa(TCNT2_7),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[0]),
        .cin(TCNT2_cout[6]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT2_7__Z.cin_used="true";
defparam TCNT2_7__Z.operation_mode="normal";
defparam TCNT2_7__Z.output_mode="reg_only";
defparam TCNT2_7__Z.lut_mask="5a5a";
defparam TCNT2_7__Z.synch_mode="off";
defparam TCNT2_7__Z.sum_lutc_input="cin";
// @13:107
  cyclone_lcell TCNT3_0__Z (
        .regout(TCNT3[0]),
        .cout(TCNT3_cout[0]),
        .clk(LCLK),
        .dataa(TCNT3[0]),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
```

```
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[1]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT3_0__Z.operation_mode="arithmetic";
defparam TCNT3_0__Z.output_mode="reg_only";
defparam TCNT3_0__Z.lut_mask="55aa";
defparam TCNT3_0__Z.synch_mode="off";
defparam TCNT3_0__Z.sum_lutc_input="datac";
// @13:107
  cyclone_lcell TCNT3_1__Z (
        .regout(TCNT3[1]),
        .cout(TCNT3_cout[1]),
        .clk(LCLK),
        .dataa(TCNT3[1]),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[1]),
        .cin(TCNT3_cout[0]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT3_1__Z.cin_used="true";
defparam TCNT3_1__Z.operation_mode="arithmetic";
defparam TCNT3_1__Z.output_mode="reg_only";
defparam TCNT3_1__Z.lut_mask="5aa0";
defparam TCNT3_1__Z.synch_mode="off";
defparam TCNT3_1__Z.sum_lutc_input="cin";
// @13:107
  cyclone_lcell TCNT3_2__Z (
        .regout(TCNT3[2]),
        .cout(TCNT3_cout[2]),
        .clk(LCLK),
        .dataa(TCNT3[2]),
```

```
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[1]),
        .cin(TCNT3_cout[1]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT3_2__Z.cin_used=”true”;
defparam TCNT3_2__Z.operation_mode=”arithmetic”;
defparam TCNT3_2__Z.output_mode=”reg_only”;
defparam TCNT3_2__Z.lut_mask=”5aa0”;
defparam TCNT3_2__Z.synch_mode=”off”;
defparam TCNT3_2__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT3_3__Z (
        .regout(TCNT3[3]),
        .cout(TCNT3_cout[3]),
        .clk(LCLK),
        .dataa(TCNT3[3]),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[1]),
        .cin(TCNT3_cout[2]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT3_3__Z.cin_used=”true”;
defparam TCNT3_3__Z.operation_mode=”arithmetic”;
defparam TCNT3_3__Z.output_mode=”reg_only”;
defparam TCNT3_3__Z.lut_mask=”5aa0”;
defparam TCNT3_3__Z.synch_mode=”off”;
defparam TCNT3_3__Z.sum_lutc_input=”cin”;
// @13:107
```

```
  cyclone_lcell TCNT3_4__Z (
        .regout(TCNT3[4]),
        .cout(TCNT3_cout[4]),
        .clk(LCLK),
        .dataa(TCNT3[4]),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[1]),
        .cin(TCNT3_cout[3]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT3_4__Z.cin_used=”true”;
defparam TCNT3_4__Z.operation_mode=”arithmetic”;
defparam TCNT3_4__Z.output_mode=”reg_only”;
defparam TCNT3_4__Z.lut_mask=”5aa0”;
defparam TCNT3_4__Z.synch_mode=”off”;
defparam TCNT3_4__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT3_5__Z (
        .regout(TCNT3[5]),
        .cout(TCNT3_cout[5]),
        .clk(LCLK),
        .dataa(TCNT3[5]),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[1]),
        .cin(TCNT3_cout[4]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT3_5__Z.cin_used=”true”;
defparam TCNT3_5__Z.operation_mode=”arithmetic”;
```

```
defparam TCNT3_5__Z.output_mode=”reg_only”;
defparam TCNT3_5__Z.lut_mask=”5aa0”;
defparam TCNT3_5__Z.synch_mode=”off”;
defparam TCNT3_5__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT3_6__Z (
        .regout(TCNT3[6]),
        .cout(TCNT3_cout[6]),
        .clk(LCLK),
        .dataa(TCNT3[6]),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[1]),
        .cin(TCNT3_cout[5]),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TCNT3_6__Z.cin_used=”true”;
defparam TCNT3_6__Z.operation_mode=”arithmetic”;
defparam TCNT3_6__Z.output_mode=”reg_only”;
defparam TCNT3_6__Z.lut_mask=”5aa0”;
defparam TCNT3_6__Z.synch_mode=”off”;
defparam TCNT3_6__Z.sum_lutc_input=”cin”;
// @13:107
  cyclone_lcell TCNT3_7__Z (
        .regout(TCNT3[7]),
        .clk(LCLK),
        .dataa(TCNT3[7]),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(TICK[1]),
        .cin(TCNT3_cout[6]),
        .inverta(GND),
        .aload(GND),
```

```
        .regcascin(GND)
);
defparam TCNT3_7__Z.cin_used=”true”;
defparam TCNT3_7__Z.operation_mode=”normal”;
defparam TCNT3_7__Z.output_mode=”reg_only”;
defparam TCNT3_7__Z.lut_mask=”5a5a”;
defparam TCNT3_7__Z.synch_mode=”off”;
defparam TCNT3_7__Z.sum_lutc_input=”cin”;
// @17:340
  cyclone_lcell TICK_2__Z (
        .regout(TICK[2]),
        .clk(LCLK),
        .dataa(un14_tcnt3_4),
        .datab(un14_tcnt3_5_0),
        .datac(un14_tcnt3_5),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TICK_2__Z.operation_mode=”normal”;
defparam TICK_2__Z.output_mode=”reg_only”;
defparam TICK_2__Z.lut_mask=”8080”;
defparam TICK_2__Z.synch_mode=”off”;
defparam TICK_2__Z.sum_lutc_input=”datac”;
// @17:340
  cyclone_lcell TICK_1__Z (
        .combout(un14_tcnt3_5),
        .regout(TICK[1]),
        .clk(LCLK),
        .dataa(un14_tcnt3_5_4),
        .datab(un14_tcnt3_5_5),
        .datac(un7_tcnt1),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
```

```
        .aload(GND),
        .regcascin(GND)
);
defparam TICK_1__Z.operation_mode=”normal”;
defparam TICK_1__Z.output_mode=”reg_and_comb”;
defparam TICK_1__Z.lut_mask=”8080”;
defparam TICK_1__Z.synch_mode=”off”;
defparam TICK_1__Z.sum_lutc_input=”datac”;
// @17:340
  cyclone_lcell TICK_0__Z (
        .combout(un7_tcnt1),
        .regout(TICK[0]),
        .clk(LCLK),
        .dataa(TCNT1[5]),
        .datab(TCNT1[4]),
        .datac(TCNT1[3]),
        .datad(un7_tcnt1_a),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TICK_0__Z.operation_mode=”normal”;
defparam TICK_0__Z.output_mode=”reg_and_comb”;
defparam TICK_0__Z.lut_mask=”0100”;
defparam TICK_0__Z.synch_mode=”off”;
defparam TICK_0__Z.sum_lutc_input=”datac”;
// @13:68
  cyclone_lcell un2_red_pulse_0_a2_x_cZ (
        .combout(un2_red_pulse_0_a2_x),
        .dataa(nLRESET),
        .datab(RED_PULSE),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
```

```
        .regcascin(GND)
);
defparam un2_red_pulse_0_a2_x_cZ.operation_mode=”normal”;
defparam un2_red_pulse_0_a2_x_cZ.output_mode=”comb_only”;
defparam un2_red_pulse_0_a2_x_cZ.lut_mask=”8888”;
defparam un2_red_pulse_0_a2_x_cZ.synch_mode=”off”;
defparam un2_red_pulse_0_a2_x_cZ.sum_lutc_input=”datac”;
// @13:89
  cyclone_lcell un2_green_pulse_0_a2_x_cZ (
        .combout(un2_green_pulse_0_a2_x),
        .dataa(nLRESET),
        .datab(GREEN_PULSE),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam un2_green_pulse_0_a2_x_cZ.operation_mode=”normal”;
defparam un2_green_pulse_0_a2_x_cZ.output_mode=”comb_only”;
defparam un2_green_pulse_0_a2_x_cZ.lut_mask=”8888”;
defparam un2_green_pulse_0_a2_x_cZ.synch_mode=”off”;
defparam un2_green_pulse_0_a2_x_cZ.sum_lutc_input=”datac”;
// @13:129
  cyclone_lcell un14_tcnt3_5_cZ (
        .combout(un14_tcnt3_5_0),
        .dataa(TCNT3[6]),
        .datab(TCNT3[7]),
        .datac(TCNT3[4]),
        .datad(TCNT3[5]),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam un14_tcnt3_5_cZ.operation_mode=”normal”;
```

```
defparam un14_tcnt3_5_cZ.output_mode=”comb_only”;
defparam un14_tcnt3_5_cZ.lut_mask=”0001”;
defparam un14_tcnt3_5_cZ.synch_mode=”off”;
defparam un14_tcnt3_5_cZ.sum_lutc_input=”datac”;
// @13:129
  cyclone_lcell un14_tcnt3_4_cZ (
        .combout(un14_tcnt3_4),
        .dataa(TCNT3[2]),
        .datab(TCNT3[3]),
        .datac(TCNT3[0]),
        .datad(TCNT3[1]),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam un14_tcnt3_4_cZ.operation_mode=”normal”;
defparam un14_tcnt3_4_cZ.output_mode=”comb_only”;
defparam un14_tcnt3_4_cZ.lut_mask=”0001”;
defparam un14_tcnt3_4_cZ.synch_mode=”off”;
defparam un14_tcnt3_4_cZ.sum_lutc_input=”datac”;
// @13:65
  cyclone_lcell nLEDR_0_0_cZ (
        .combout(nLEDR_0_0),
        .dataa(TICK[2]),
        .datab(TMONOSR),
        .datac(nLEDR),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam nLEDR_0_0_cZ.operation_mode=”normal”;
defparam nLEDR_0_0_cZ.output_mode=”comb_only”;
defparam nLEDR_0_0_cZ.lut_mask=”f2f2”;
defparam nLEDR_0_0_cZ.synch_mode=”off”;
```

```
defparam nLEDR_0_0_cZ.sum_lutc_input=”datac”;
// @13:86
  cyclone_lcell nLEDG_0_0_cZ (
        .combout(nLEDG_0_0),
        .dataa(TICK[2]),
        .datab(TMONOSG),
        .datac(nLEDG),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam nLEDG_0_0_cZ.operation_mode=”normal”;
defparam nLEDG_0_0_cZ.output_mode=”comb_only”;
defparam nLEDG_0_0_cZ.lut_mask=”f2f2”;
defparam nLEDG_0_0_cZ.synch_mode=”off”;
defparam nLEDG_0_0_cZ.sum_lutc_input=”datac”;
// @13:65
  cyclone_lcell TMONOSR_0_0_a2_cZ (
        .combout(TMONOSR_0_0_a2),
        .dataa(TMONOSR),
        .datab(TICK[2]),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TMONOSR_0_0_a2_cZ.operation_mode=”normal”;
defparam TMONOSR_0_0_a2_cZ.output_mode=”comb_only”;
defparam TMONOSR_0_0_a2_cZ.lut_mask=”2222”;
defparam TMONOSR_0_0_a2_cZ.synch_mode=”off”;
defparam TMONOSR_0_0_a2_cZ.sum_lutc_input=”datac”;
// @13:86
  cyclone_lcell TMONOSG_0_0_a2_cZ (
```

```
        .combout(TMONOSG_0_0_a2),
        .dataa(TMONOSG),
        .datab(TICK[2]),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam TMONOSG_0_0_a2_cZ.operation_mode=”normal”;
defparam TMONOSG_0_0_a2_cZ.output_mode=”comb_only”;
defparam TMONOSG_0_0_a2_cZ.lut_mask=”2222”;
defparam TMONOSG_0_0_a2_cZ.synch_mode=”off”;
defparam TMONOSG_0_0_a2_cZ.sum_lutc_input=”datac”;
// @13:86
  cyclone_lcell nLEDG_Z (
        .regout(nLEDG),
        .clk(LCLK),
        .dataa(nLEDG_0_0),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(un2_green_pulse_0_a2_x),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(nLRESET_i_x_i),
        .regcascin(GND)
);
defparam nLEDG_Z.operation_mode=”normal”;
defparam nLEDG_Z.output_mode=”reg_only”;
defparam nLEDG_Z.lut_mask=”aaaa”;
defparam nLEDG_Z.synch_mode=”off”;
defparam nLEDG_Z.sum_lutc_input=”datac”;
// @13:65
  cyclone_lcell nLEDR_Z (
        .regout(nLEDR),
        .clk(LCLK),
```

```
        .dataa(nLEDR_0_0),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(un2_red_pulse_0_a2_x),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(nLRESET_i_x_i),
        .regcascin(GND)
);
defparam nLEDR_Z.operation_mode=”normal”;
defparam nLEDR_Z.output_mode=”reg_only”;
defparam nLEDR_Z.lut_mask=”aaaa”;
defparam nLEDR_Z.synch_mode=”off”;
defparam nLEDR_Z.sum_lutc_input=”datac”;
// @13:86
  cyclone_lcell TMONOSG_Z (
        .regout(TMONOSG),
        .clk(LCLK),
        .dataa(TMONOSG_0_0_a2),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GREEN_PULSE),
        .regcascin(GND)
);
defparam TMONOSG_Z.operation_mode=”normal”;
defparam TMONOSG_Z.output_mode=”reg_only”;
defparam TMONOSG_Z.lut_mask=”aaaa”;
defparam TMONOSG_Z.synch_mode=”off”;
defparam TMONOSG_Z.sum_lutc_input=”datac”;
// @13:65
  cyclone_lcell TMONOSR_Z (
        .regout(TMONOSR),
        .clk(LCLK),
        .dataa(TMONOSR_0_0_a2),
```

```
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(RED_PULSE),
        .regcascin(GND)
);
defparam TMONOSR_Z.operation_mode=”normal”;
defparam TMONOSR_Z.output_mode=”reg_only”;
defparam TMONOSR_Z.lut_mask=”aaaa”;
defparam TMONOSR_Z.synch_mode=”off”;
defparam TMONOSR_Z.sum_lutc_input=”datac”;
  assign  nLRESET_i = ~ nLRESET;
endmodule /* led_if */

// VQM4.1+
module localbusif (
  REG_DOUT_15,
  REG_DOUT_14,
  REG_DOUT_13,
  REG_DOUT_12,
  REG_DOUT_11,
  REG_DOUT_10,
  REG_DOUT_9,
  REG_DOUT_8,
  REG_DOUT_7,
  REG_DOUT_6,
  REG_DOUT_5,
  REG_DOUT_4,
  REG_DOUT_3,
  REG_DOUT_2,
  REG_DOUT_1,
  REG_DOUT_0,
  LAD_OUT_0_a4_x_15,
  LAD_OUT_0_a4_x_14,
  LAD_OUT_0_a4_x_13,
  LAD_OUT_0_a4_x_12,
  LAD_OUT_0_a4_x_11,
  LAD_OUT_0_a4_x_10,
```

LAD_OUT_0_a4_x_9,
LAD_OUT_0_a4_x_8,
LAD_OUT_0_a4_x_7,
LAD_OUT_0_a4_x_6,
LAD_OUT_0_a4_x_5,
LAD_OUT_0_a4_x_4,
LAD_OUT_0_a4_x_3,
LAD_OUT_0_a4_x_2,
LAD_OUT_0_a4_x_1,
LAD_OUT_0_a4_x_0,
LAD_IN_15,
LAD_IN_14,
LAD_IN_13,
LAD_IN_12,
LAD_IN_11,
LAD_IN_10,
LAD_IN_9,
LAD_IN_8,
LAD_IN_7,
LAD_IN_6,
LAD_IN_5,
LAD_IN_4,
LAD_IN_3,
LAD_IN_2,
LAD_IN_1,
LAD_IN_0,
REG_ADDR_i_m2_x_15,
REG_ADDR_i_m2_x_14,
REG_ADDR_i_m2_x_13,
REG_ADDR_i_m2_x_12,
REG_ADDR_i_m2_x_11,
REG_ADDR_i_m2_x_10,
REG_ADDR_i_m2_x_9,
REG_ADDR_i_m2_x_8,
REG_ADDR_i_m2_x_7,
REG_ADDR_i_m2_x_6,
REG_ADDR_i_m2_x_5,
REG_ADDR_i_m2_x_4,
REG_ADDR_i_m2_x_3,
REG_ADDR_i_m2_x_2,
REG_ADDR_i_m2_x_1,
REG_ADDR_i_m2_x_0,
REG_RDEN_i_x,

```
  nREADY_s_i_i_x,
  un2_lad_oe_i_0_a2_x,
  WnR,
  WREN_s,
  nBLAST,
  nLRESET,
  nADS,
  LCLK
);
input REG_DOUT_15 ;
input REG_DOUT_14 ;
input REG_DOUT_13 ;
input REG_DOUT_12 ;
input REG_DOUT_11 ;
input REG_DOUT_10 ;
input REG_DOUT_9 ;
input REG_DOUT_8 ;
input REG_DOUT_7 ;
input REG_DOUT_6 ;
input REG_DOUT_5 ;
input REG_DOUT_4 ;
input REG_DOUT_3 ;
input REG_DOUT_2 ;
input REG_DOUT_1 ;
input REG_DOUT_0 ;
output LAD_OUT_0_a4_x_15 ;
output LAD_OUT_0_a4_x_14 ;
output LAD_OUT_0_a4_x_13 ;
output LAD_OUT_0_a4_x_12 ;
output LAD_OUT_0_a4_x_11 ;
output LAD_OUT_0_a4_x_10 ;
output LAD_OUT_0_a4_x_9 ;
output LAD_OUT_0_a4_x_8 ;
output LAD_OUT_0_a4_x_7 ;
output LAD_OUT_0_a4_x_6 ;
output LAD_OUT_0_a4_x_5 ;
output LAD_OUT_0_a4_x_4 ;
output LAD_OUT_0_a4_x_3 ;
output LAD_OUT_0_a4_x_2 ;
output LAD_OUT_0_a4_x_1 ;
output LAD_OUT_0_a4_x_0 ;
input LAD_IN_15 ;
input LAD_IN_14 ;
```

```
input LAD_IN_13 ;
input LAD_IN_12 ;
input LAD_IN_11 ;
input LAD_IN_10 ;
input LAD_IN_9 ;
input LAD_IN_8 ;
input LAD_IN_7 ;
input LAD_IN_6 ;
input LAD_IN_5 ;
input LAD_IN_4 ;
input LAD_IN_3 ;
input LAD_IN_2 ;
input LAD_IN_1 ;
input LAD_IN_0 ;
output REG_ADDR_i_m2_x_15 ;
output REG_ADDR_i_m2_x_14 ;
output REG_ADDR_i_m2_x_13 ;
output REG_ADDR_i_m2_x_12 ;
output REG_ADDR_i_m2_x_11 ;
output REG_ADDR_i_m2_x_10 ;
output REG_ADDR_i_m2_x_9 ;
output REG_ADDR_i_m2_x_8 ;
output REG_ADDR_i_m2_x_7 ;
output REG_ADDR_i_m2_x_6 ;
output REG_ADDR_i_m2_x_5 ;
output REG_ADDR_i_m2_x_4 ;
output REG_ADDR_i_m2_x_3 ;
output REG_ADDR_i_m2_x_2 ;
output REG_ADDR_i_m2_x_1 ;
output REG_ADDR_i_m2_x_0 ;
output REG_RDEN_i_x ;
output nREADY_s_i_i_x ;
output un2_lad_oe_i_0_a2_x ;
input WnR ;
output WREN_s ;
input nBLAST ;
input nLRESET ;
input nADS ;
input LCLK ;
wire REG_DOUT_15 ;
wire REG_DOUT_14 ;
wire REG_DOUT_13 ;
wire REG_DOUT_12 ;
```

```
wire REG_DOUT_11 ;
wire REG_DOUT_10 ;
wire REG_DOUT_9 ;
wire REG_DOUT_8 ;
wire REG_DOUT_7 ;
wire REG_DOUT_6 ;
wire REG_DOUT_5 ;
wire REG_DOUT_4 ;
wire REG_DOUT_3 ;
wire REG_DOUT_2 ;
wire REG_DOUT_1 ;
wire REG_DOUT_0 ;
wire LAD_OUT_0_a4_x_15 ;
wire LAD_OUT_0_a4_x_14 ;
wire LAD_OUT_0_a4_x_13 ;
wire LAD_OUT_0_a4_x_12 ;
wire LAD_OUT_0_a4_x_11 ;
wire LAD_OUT_0_a4_x_10 ;
wire LAD_OUT_0_a4_x_9 ;
wire LAD_OUT_0_a4_x_8 ;
wire LAD_OUT_0_a4_x_7 ;
wire LAD_OUT_0_a4_x_6 ;
wire LAD_OUT_0_a4_x_5 ;
wire LAD_OUT_0_a4_x_4 ;
wire LAD_OUT_0_a4_x_3 ;
wire LAD_OUT_0_a4_x_2 ;
wire LAD_OUT_0_a4_x_1 ;
wire LAD_OUT_0_a4_x_0 ;
wire LAD_IN_15 ;
wire LAD_IN_14 ;
wire LAD_IN_13 ;
wire LAD_IN_12 ;
wire LAD_IN_11 ;
wire LAD_IN_10 ;
wire LAD_IN_9 ;
wire LAD_IN_8 ;
wire LAD_IN_7 ;
wire LAD_IN_6 ;
wire LAD_IN_5 ;
wire LAD_IN_4 ;
wire LAD_IN_3 ;
wire LAD_IN_2 ;
wire LAD_IN_1 ;
```

```
wire LAD_IN_0 ;
wire REG_ADDR_i_m2_x_15 ;
wire REG_ADDR_i_m2_x_14 ;
wire REG_ADDR_i_m2_x_13 ;
wire REG_ADDR_i_m2_x_12 ;
wire REG_ADDR_i_m2_x_11 ;
wire REG_ADDR_i_m2_x_10 ;
wire REG_ADDR_i_m2_x_9 ;
wire REG_ADDR_i_m2_x_8 ;
wire REG_ADDR_i_m2_x_7 ;
wire REG_ADDR_i_m2_x_6 ;
wire REG_ADDR_i_m2_x_5 ;
wire REG_ADDR_i_m2_x_4 ;
wire REG_ADDR_i_m2_x_3 ;
wire REG_ADDR_i_m2_x_2 ;
wire REG_ADDR_i_m2_x_1 ;
wire REG_ADDR_i_m2_x_0 ;
wire REG_RDEN_i_x ;
wire nREADY_s_i_i_x ;
wire un2_lad_oe_i_0_a2_x ;
wire WnR ;
wire WREN_s ;
wire nBLAST ;
wire nLRESET ;
wire nADS ;
wire LCLK ;
wire [1:0] LBSTATE;
wire VCC ;
wire ADDR_s_0_sqmuxa_0_a2_0_a2_x ;
wire m9_0_a_x ;
wire m5_0_a_x ;
wire RDEN_s ;
wire nREADY_s_i ;
wire N_36 ;
wire N_35 ;
wire N_34 ;
wire N_33 ;
wire GND ;
wire nLRESET_i ;
//@1:1
  assign VCC = 1'b1;
  assign GND = 1'b0;
// @14:83
```

```
  cyclone_lcell ADDR_s_0__Z (
        .combout(REG_ADDR_i_m2_x_0),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_0),
        .datac(LAD_IN_0),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_0__Z.operation_mode="normal";
defparam ADDR_s_0__Z.output_mode="comb_only";
defparam ADDR_s_0__Z.lut_mask="e4e4";
defparam ADDR_s_0__Z.synch_mode="on";
defparam ADDR_s_0__Z.sum_lutc_input="qfbk";
// @14:83
  cyclone_lcell ADDR_s_1__Z (
        .combout(REG_ADDR_i_m2_x_1),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_1),
        .datac(LAD_IN_1),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_1__Z.operation_mode="normal";
defparam ADDR_s_1__Z.output_mode="comb_only";
defparam ADDR_s_1__Z.lut_mask="e4e4";
defparam ADDR_s_1__Z.synch_mode="on";
defparam ADDR_s_1__Z.sum_lutc_input="qfbk";
// @14:83
  cyclone_lcell ADDR_s_2__Z (
```

```
        .combout(REG_ADDR_i_m2_x_2),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_2),
        .datac(LAD_IN_2),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_2__Z.operation_mode=”normal”;
defparam ADDR_s_2__Z.output_mode=”comb_only”;
defparam ADDR_s_2__Z.lut_mask=”e4e4”;
defparam ADDR_s_2__Z.synch_mode=”on”;
defparam ADDR_s_2__Z.sum_lutc_input=”qfbk”;
// @14:83
  cyclone_lcell ADDR_s_3__Z (
        .combout(REG_ADDR_i_m2_x_3),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_3),
        .datac(LAD_IN_3),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_3__Z.operation_mode=”normal”;
defparam ADDR_s_3__Z.output_mode=”comb_only”;
defparam ADDR_s_3__Z.lut_mask=”e4e4”;
defparam ADDR_s_3__Z.synch_mode=”on”;
defparam ADDR_s_3__Z.sum_lutc_input=”qfbk”;
// @14:83
  cyclone_lcell ADDR_s_4__Z (
        .combout(REG_ADDR_i_m2_x_4),
```

```
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_4),
        .datac(LAD_IN_4),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_4__Z.operation_mode=”normal”;
defparam ADDR_s_4__Z.output_mode=”comb_only”;
defparam ADDR_s_4__Z.lut_mask=”e4e4”;
defparam ADDR_s_4__Z.synch_mode=”on”;
defparam ADDR_s_4__Z.sum_lutc_input=”qfbk”;
// @14:83
  cyclone_lcell ADDR_s_5__Z (
        .combout(REG_ADDR_i_m2_x_5),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_5),
        .datac(LAD_IN_5),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_5__Z.operation_mode=”normal”;
defparam ADDR_s_5__Z.output_mode=”comb_only”;
defparam ADDR_s_5__Z.lut_mask=”e4e4”;
defparam ADDR_s_5__Z.synch_mode=”on”;
defparam ADDR_s_5__Z.sum_lutc_input=”qfbk”;
// @14:83
  cyclone_lcell ADDR_s_6__Z (
        .combout(REG_ADDR_i_m2_x_6),
        .clk(LCLK),
```

```
        .dataa(nADS),
        .datab(LAD_IN_6),
        .datac(LAD_IN_6),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_6__Z.operation_mode="normal";
defparam ADDR_s_6__Z.output_mode="comb_only";
defparam ADDR_s_6__Z.lut_mask="e4e4";
defparam ADDR_s_6__Z.synch_mode="on";
defparam ADDR_s_6__Z.sum_lutc_input="qfbk";
// @14:83
  cyclone_lcell ADDR_s_7__Z (
        .combout(REG_ADDR_i_m2_x_7),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_7),
        .datac(LAD_IN_7),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_7__Z.operation_mode="normal";
defparam ADDR_s_7__Z.output_mode="comb_only";
defparam ADDR_s_7__Z.lut_mask="e4e4";
defparam ADDR_s_7__Z.synch_mode="on";
defparam ADDR_s_7__Z.sum_lutc_input="qfbk";
// @14:83
  cyclone_lcell ADDR_s_8__Z (
        .combout(REG_ADDR_i_m2_x_8),
        .clk(LCLK),
        .dataa(nADS),
```

```
        .datab(LAD_IN_8),
        .datac(LAD_IN_8),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_8__Z.operation_mode=”normal”;
defparam ADDR_s_8__Z.output_mode=”comb_only”;
defparam ADDR_s_8__Z.lut_mask=”e4e4”;
defparam ADDR_s_8__Z.synch_mode=”on”;
defparam ADDR_s_8__Z.sum_lutc_input=”qfbk”;
// @14:83
  cyclone_lcell ADDR_s_9__Z (
        .combout(REG_ADDR_i_m2_x_9),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_9),
        .datac(LAD_IN_9),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_9__Z.operation_mode=”normal”;
defparam ADDR_s_9__Z.output_mode=”comb_only”;
defparam ADDR_s_9__Z.lut_mask=”e4e4”;
defparam ADDR_s_9__Z.synch_mode=”on”;
defparam ADDR_s_9__Z.sum_lutc_input=”qfbk”;
// @14:83
  cyclone_lcell ADDR_s_10__Z (
        .combout(REG_ADDR_i_m2_x_10),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_10),
```

```
        .datac(LAD_IN_10),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_10__Z.operation_mode=”normal”;
defparam ADDR_s_10__Z.output_mode=”comb_only”;
defparam ADDR_s_10__Z.lut_mask=”e4e4”;
defparam ADDR_s_10__Z.synch_mode=”on”;
defparam ADDR_s_10__Z.sum_lutc_input=”qfbk”;
// @14:83
  cyclone_lcell ADDR_s_11__Z (
        .combout(REG_ADDR_i_m2_x_11),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_11),
        .datac(LAD_IN_11),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_11__Z.operation_mode=”normal”;
defparam ADDR_s_11__Z.output_mode=”comb_only”;
defparam ADDR_s_11__Z.lut_mask=”e4e4”;
defparam ADDR_s_11__Z.synch_mode=”on”;
defparam ADDR_s_11__Z.sum_lutc_input=”qfbk”;
// @14:83
  cyclone_lcell ADDR_s_12__Z (
        .combout(REG_ADDR_i_m2_x_12),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_12),
        .datac(LAD_IN_12),
```

```
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_12__Z.operation_mode="normal";
defparam ADDR_s_12__Z.output_mode="comb_only";
defparam ADDR_s_12__Z.lut_mask="e4e4";
defparam ADDR_s_12__Z.synch_mode="on";
defparam ADDR_s_12__Z.sum_lutc_input="qfbk";
// @14:83
  cyclone_lcell ADDR_s_13__Z (
        .combout(REG_ADDR_i_m2_x_13),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_13),
        .datac(LAD_IN_13),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_13__Z.operation_mode="normal";
defparam ADDR_s_13__Z.output_mode="comb_only";
defparam ADDR_s_13__Z.lut_mask="e4e4";
defparam ADDR_s_13__Z.synch_mode="on";
defparam ADDR_s_13__Z.sum_lutc_input="qfbk";
// @14:83
  cyclone_lcell ADDR_s_14__Z (
        .combout(REG_ADDR_i_m2_x_14),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_14),
        .datac(LAD_IN_14),
        .datad(VCC),
```

```
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_14__Z.operation_mode="normal";
defparam ADDR_s_14__Z.output_mode="comb_only";
defparam ADDR_s_14__Z.lut_mask="e4e4";
defparam ADDR_s_14__Z.synch_mode="on";
defparam ADDR_s_14__Z.sum_lutc_input="qfbk";
// @14:83
  cyclone_lcell ADDR_s_15__Z (
        .combout(REG_ADDR_i_m2_x_15),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LAD_IN_15),
        .datac(LAD_IN_15),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(VCC),
        .ena(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam ADDR_s_15__Z.operation_mode="normal";
defparam ADDR_s_15__Z.output_mode="comb_only";
defparam ADDR_s_15__Z.lut_mask="e4e4";
defparam ADDR_s_15__Z.synch_mode="on";
defparam ADDR_s_15__Z.sum_lutc_input="qfbk";
// @14:83
  cyclone_lcell LBSTATE_1__Z (
        .regout(LBSTATE[1]),
        .clk(LCLK),
        .dataa(nBLAST),
        .datab(m9_0_a_x),
        .datac(LBSTATE[1]),
        .datad(LBSTATE[0]),
        .aclr(nLRESET_i),
```

```
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LBSTATE_1__Z.operation_mode=”normal”;
defparam LBSTATE_1__Z.output_mode=”reg_only”;
defparam LBSTATE_1__Z.lut_mask=”05fc”;
defparam LBSTATE_1__Z.synch_mode=”off”;
defparam LBSTATE_1__Z.sum_lutc_input=”datac”;
// @14:83
  cyclone_lcell LBSTATE_0__Z (
        .regout(LBSTATE[0]),
        .clk(LCLK),
        .dataa(nBLAST),
        .datab(m5_0_a_x),
        .datac(LBSTATE[0]),
        .datad(LBSTATE[1]),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LBSTATE_0__Z.operation_mode=”normal”;
defparam LBSTATE_0__Z.output_mode=”reg_only”;
defparam LBSTATE_0__Z.lut_mask=”05fc”;
defparam LBSTATE_0__Z.synch_mode=”off”;
defparam LBSTATE_0__Z.sum_lutc_input=”datac”;
// @14:83
  cyclone_lcell WREN_s_Z (
        .regout(WREN_s),
        .clk(LCLK),
        .dataa(WnR),
        .datab(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
```

```
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam WREN_s_Z.operation_mode=”normal”;
defparam WREN_s_Z.output_mode=”reg_only”;
defparam WREN_s_Z.lut_mask=”8888”;
defparam WREN_s_Z.synch_mode=”off”;
defparam WREN_s_Z.sum_lutc_input=”datac”;
// @14:83
  cyclone_lcell RDEN_s_Z (
        .regout(RDEN_s),
        .clk(LCLK),
        .dataa(WnR),
        .datab(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam RDEN_s_Z.operation_mode=”normal”;
defparam RDEN_s_Z.output_mode=”reg_only”;
defparam RDEN_s_Z.lut_mask=”4444”;
defparam RDEN_s_Z.synch_mode=”off”;
defparam RDEN_s_Z.sum_lutc_input=”datac”;
// @14:83
  cyclone_lcell nREADY_s_i_Z (
        .combout(ADDR_s_0_sqmuxa_0_a2_0_a2_x),
        .regout(nREADY_s_i),
        .clk(LCLK),
        .dataa(nADS),
        .datab(LBSTATE[0]),
        .datac(LBSTATE[1]),
        .datad(VCC),
        .aclr(nLRESET_i),
        .sclr(GND),
```

```
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam nREADY_s_i_Z.operation_mode=”normal”;
defparam nREADY_s_i_Z.output_mode=”reg_and_comb”;
defparam nREADY_s_i_Z.lut_mask=”0101”;
defparam nREADY_s_i_Z.synch_mode=”off”;
defparam nREADY_s_i_Z.sum_lutc_input=”datac”;
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_0_ (
        .combout(LAD_OUT_0_a4_x_0),
        .dataa(REG_DOUT_0),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_0_.operation_mode=”normal”;
defparam LAD_OUT_0_a4_x_0_.output_mode=”comb_only”;
defparam LAD_OUT_0_a4_x_0_.lut_mask=”8888”;
defparam LAD_OUT_0_a4_x_0_.synch_mode=”off”;
defparam LAD_OUT_0_a4_x_0_.sum_lutc_input=”datac”;
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_1_ (
        .combout(LAD_OUT_0_a4_x_1),
        .dataa(REG_DOUT_1),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
```

```
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_1_.operation_mode=”normal”;
defparam LAD_OUT_0_a4_x_1_.output_mode=”comb_only”;
defparam LAD_OUT_0_a4_x_1_.lut_mask=”8888”;
defparam LAD_OUT_0_a4_x_1_.synch_mode=”off”;
defparam LAD_OUT_0_a4_x_1_.sum_lutc_input=”datac”;
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_2_ (
        .combout(LAD_OUT_0_a4_x_2),
        .dataa(REG_DOUT_2),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_2_.operation_mode=”normal”;
defparam LAD_OUT_0_a4_x_2_.output_mode=”comb_only”;
defparam LAD_OUT_0_a4_x_2_.lut_mask=”8888”;
defparam LAD_OUT_0_a4_x_2_.synch_mode=”off”;
defparam LAD_OUT_0_a4_x_2_.sum_lutc_input=”datac”;
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_3_ (
        .combout(LAD_OUT_0_a4_x_3),
        .dataa(REG_DOUT_3),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
```

```
defparam LAD_OUT_0_a4_x_3_.operation_mode=”normal”;
defparam LAD_OUT_0_a4_x_3_.output_mode=”comb_only”;
defparam LAD_OUT_0_a4_x_3_.lut_mask=”8888”;
defparam LAD_OUT_0_a4_x_3_.synch_mode=”off”;
defparam LAD_OUT_0_a4_x_3_.sum_lutc_input=”datac”;
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_4_ (
        .combout(LAD_OUT_0_a4_x_4),
        .dataa(REG_DOUT_4),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_4_.operation_mode=”normal”;
defparam LAD_OUT_0_a4_x_4_.output_mode=”comb_only”;
defparam LAD_OUT_0_a4_x_4_.lut_mask=”8888”;
defparam LAD_OUT_0_a4_x_4_.synch_mode=”off”;
defparam LAD_OUT_0_a4_x_4_.sum_lutc_input=”datac”;
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_5_ (
        .combout(LAD_OUT_0_a4_x_5),
        .dataa(REG_DOUT_5),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_5_.operation_mode=”normal”;
defparam LAD_OUT_0_a4_x_5_.output_mode=”comb_only”;
defparam LAD_OUT_0_a4_x_5_.lut_mask=”8888”;
```

```
defparam LAD_OUT_0_a4_x_5_.synch_mode=”off”;
defparam LAD_OUT_0_a4_x_5_.sum_lutc_input=”datac”;
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_6_ (
        .combout(LAD_OUT_0_a4_x_6),
        .dataa(REG_DOUT_6),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_6_.operation_mode=”normal”;
defparam LAD_OUT_0_a4_x_6_.output_mode=”comb_only”;
defparam LAD_OUT_0_a4_x_6_.lut_mask=”8888”;
defparam LAD_OUT_0_a4_x_6_.synch_mode=”off”;
defparam LAD_OUT_0_a4_x_6_.sum_lutc_input=”datac”;
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_7_ (
        .combout(LAD_OUT_0_a4_x_7),
        .dataa(REG_DOUT_7),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_7_.operation_mode=”normal”;
defparam LAD_OUT_0_a4_x_7_.output_mode=”comb_only”;
defparam LAD_OUT_0_a4_x_7_.lut_mask=”8888”;
defparam LAD_OUT_0_a4_x_7_.synch_mode=”off”;
defparam LAD_OUT_0_a4_x_7_.sum_lutc_input=”datac”;
// @14:70
```

```
  cyclone_lcell LAD_OUT_0_a4_x_8_ (
        .combout(LAD_OUT_0_a4_x_8),
        .dataa(REG_DOUT_8),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_8_.operation_mode=”normal”;
defparam LAD_OUT_0_a4_x_8_.output_mode=”comb_only”;
defparam LAD_OUT_0_a4_x_8_.lut_mask=”8888”;
defparam LAD_OUT_0_a4_x_8_.synch_mode=”off”;
defparam LAD_OUT_0_a4_x_8_.sum_lutc_input=”datac”;
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_9_ (
        .combout(LAD_OUT_0_a4_x_9),
        .dataa(REG_DOUT_9),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_9_.operation_mode=”normal”;
defparam LAD_OUT_0_a4_x_9_.output_mode=”comb_only”;
defparam LAD_OUT_0_a4_x_9_.lut_mask=”8888”;
defparam LAD_OUT_0_a4_x_9_.synch_mode=”off”;
defparam LAD_OUT_0_a4_x_9_.sum_lutc_input=”datac”;
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_10_ (
        .combout(LAD_OUT_0_a4_x_10),
        .dataa(REG_DOUT_10),
```

```
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_10_.operation_mode=”normal”;
defparam LAD_OUT_0_a4_x_10_.output_mode=”comb_only”;
defparam LAD_OUT_0_a4_x_10_.lut_mask=”8888”;
defparam LAD_OUT_0_a4_x_10_.synch_mode=”off”;
defparam LAD_OUT_0_a4_x_10_.sum_lutc_input=”datac”;
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_11_ (
        .combout(LAD_OUT_0_a4_x_11),
        .dataa(REG_DOUT_11),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_11_.operation_mode=”normal”;
defparam LAD_OUT_0_a4_x_11_.output_mode=”comb_only”;
defparam LAD_OUT_0_a4_x_11_.lut_mask=”8888”;
defparam LAD_OUT_0_a4_x_11_.synch_mode=”off”;
defparam LAD_OUT_0_a4_x_11_.sum_lutc_input=”datac”;
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_12_ (
        .combout(LAD_OUT_0_a4_x_12),
        .dataa(REG_DOUT_12),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
```

```
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_12_.operation_mode="normal";
defparam LAD_OUT_0_a4_x_12_.output_mode="comb_only";
defparam LAD_OUT_0_a4_x_12_.lut_mask="8888";
defparam LAD_OUT_0_a4_x_12_.synch_mode="off";
defparam LAD_OUT_0_a4_x_12_.sum_lutc_input="datac";
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_13_ (
        .combout(LAD_OUT_0_a4_x_13),
        .dataa(REG_DOUT_13),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_13_.operation_mode="normal";
defparam LAD_OUT_0_a4_x_13_.output_mode="comb_only";
defparam LAD_OUT_0_a4_x_13_.lut_mask="8888";
defparam LAD_OUT_0_a4_x_13_.synch_mode="off";
defparam LAD_OUT_0_a4_x_13_.sum_lutc_input="datac";
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_14_ (
        .combout(LAD_OUT_0_a4_x_14),
        .dataa(REG_DOUT_14),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
```

```
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_14_.operation_mode="normal";
defparam LAD_OUT_0_a4_x_14_.output_mode="comb_only";
defparam LAD_OUT_0_a4_x_14_.lut_mask="8888";
defparam LAD_OUT_0_a4_x_14_.synch_mode="off";
defparam LAD_OUT_0_a4_x_14_.sum_lutc_input="datac";
// @14:70
  cyclone_lcell LAD_OUT_0_a4_x_15_ (
        .combout(LAD_OUT_0_a4_x_15),
        .dataa(REG_DOUT_15),
        .datab(un2_lad_oe_i_0_a2_x),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam LAD_OUT_0_a4_x_15_.operation_mode="normal";
defparam LAD_OUT_0_a4_x_15_.output_mode="comb_only";
defparam LAD_OUT_0_a4_x_15_.lut_mask="8888";
defparam LAD_OUT_0_a4_x_15_.synch_mode="off";
defparam LAD_OUT_0_a4_x_15_.sum_lutc_input="datac";
// @14:69
  cyclone_lcell un2_lad_oe_i_0_a2_x_cZ (
        .combout(un2_lad_oe_i_0_a2_x),
        .dataa(nADS),
        .datab(RDEN_s),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
```

```
        .regcascin(GND)
);
defparam un2_lad_oe_i_0_a2_x_cZ.operation_mode=”normal”;
defparam un2_lad_oe_i_0_a2_x_cZ.output_mode=”comb_only”;
defparam un2_lad_oe_i_0_a2_x_cZ.lut_mask=”8888”;
defparam un2_lad_oe_i_0_a2_x_cZ.synch_mode=”off”;
defparam un2_lad_oe_i_0_a2_x_cZ.sum_lutc_input=”datac”;
  cyclone_lcell nREADY_s_i_i_x_cZ (
        .combout(nREADY_s_i_i_x),
        .dataa(nREADY_s_i),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam nREADY_s_i_i_x_cZ.operation_mode=”normal”;
defparam nREADY_s_i_i_x_cZ.output_mode=”comb_only”;
defparam nREADY_s_i_i_x_cZ.lut_mask=”5555”;
defparam nREADY_s_i_i_x_cZ.synch_mode=”off”;
defparam nREADY_s_i_i_x_cZ.sum_lutc_input=”datac”;
// @14:75
  cyclone_lcell REG_RDEN_i_x_cZ (
        .combout(REG_RDEN_i_x),
        .dataa(RDEN_s),
        .datab(nADS),
        .datac(WnR),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam REG_RDEN_i_x_cZ.operation_mode=”normal”;
defparam REG_RDEN_i_x_cZ.output_mode=”comb_only”;
```

```
defparam REG_RDEN_i_x_cZ.lut_mask=”0b0b”;
defparam REG_RDEN_i_x_cZ.synch_mode=”off”;
defparam REG_RDEN_i_x_cZ.sum_lutc_input=”datac”;
// @14:83
  cyclone_lcell m9_0_a_x_cZ (
        .combout(m9_0_a_x),
        .dataa(nADS),
        .datab(WnR),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam m9_0_a_x_cZ.operation_mode=”normal”;
defparam m9_0_a_x_cZ.output_mode=”comb_only”;
defparam m9_0_a_x_cZ.lut_mask=”1111”;
defparam m9_0_a_x_cZ.synch_mode=”off”;
defparam m9_0_a_x_cZ.sum_lutc_input=”datac”;
// @14:83
  cyclone_lcell m5_0_a_x_cZ (
        .combout(m5_0_a_x),
        .dataa(WnR),
        .datab(nADS),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam m5_0_a_x_cZ.operation_mode=”normal”;
defparam m5_0_a_x_cZ.output_mode=”comb_only”;
defparam m5_0_a_x_cZ.lut_mask=”2222”;
defparam m5_0_a_x_cZ.synch_mode=”off”;
defparam m5_0_a_x_cZ.sum_lutc_input=”datac”;
```

```
//@14:83
  assign  nLRESET_i = ~ nLRESET;
endmodule /* localbusif */

// VQM4.1+
module pdl_if (
  PDL_WRITE_7,
  PDL_WRITE_6,
  PDL_WRITE_5,
  PDL_WRITE_4,
  PDL_WRITE_3,
  PDL_WRITE_2,
  PDL_WRITE_1,
  PDL_WRITE_0,
  DDLY_OUT_0_a2_x_7,
  DDLY_OUT_0_a2_x_6,
  DDLY_OUT_0_a2_x_5,
  DDLY_OUT_0_a2_x_4,
  DDLY_OUT_0_a2_x_3,
  DDLY_OUT_0_a2_x_2,
  DDLY_OUT_0_a2_x_1,
  DDLY_OUT_0_a2_x_0,
  DDLY_OE
);
input PDL_WRITE_7 ;
input PDL_WRITE_6 ;
input PDL_WRITE_5 ;
input PDL_WRITE_4 ;
input PDL_WRITE_3 ;
input PDL_WRITE_2 ;
input PDL_WRITE_1 ;
input PDL_WRITE_0 ;
output DDLY_OUT_0_a2_x_7 ;
output DDLY_OUT_0_a2_x_6 ;
output DDLY_OUT_0_a2_x_5 ;
output DDLY_OUT_0_a2_x_4 ;
output DDLY_OUT_0_a2_x_3 ;
output DDLY_OUT_0_a2_x_2 ;
output DDLY_OUT_0_a2_x_1 ;
output DDLY_OUT_0_a2_x_0 ;
input DDLY_OE ;
wire PDL_WRITE_7 ;
wire PDL_WRITE_6 ;
```

```
wire PDL_WRITE_5 ;
wire PDL_WRITE_4 ;
wire PDL_WRITE_3 ;
wire PDL_WRITE_2 ;
wire PDL_WRITE_1 ;
wire PDL_WRITE_0 ;
wire DDLY_OUT_0_a2_x_7 ;
wire DDLY_OUT_0_a2_x_6 ;
wire DDLY_OUT_0_a2_x_5 ;
wire DDLY_OUT_0_a2_x_4 ;
wire DDLY_OUT_0_a2_x_3 ;
wire DDLY_OUT_0_a2_x_2 ;
wire DDLY_OUT_0_a2_x_1 ;
wire DDLY_OUT_0_a2_x_0 ;
wire DDLY_OE ;
wire GND ;
wire VCC ;
  assign VCC = 1'b1;
  assign GND = 1'b0;
// @15:62
  cyclone_lcell DDLY_OUT_0_a2_x_0_ (
        .combout(DDLY_OUT_0_a2_x_0),
        .dataa(PDL_WRITE_0),
        .datab(DDLY_OE),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam DDLY_OUT_0_a2_x_0_.operation_mode="normal";
defparam DDLY_OUT_0_a2_x_0_.output_mode="comb_only";
defparam DDLY_OUT_0_a2_x_0_.lut_mask="8888";
defparam DDLY_OUT_0_a2_x_0_.synch_mode="off";
defparam DDLY_OUT_0_a2_x_0_.sum_lutc_input="datac";
// @15:62
  cyclone_lcell DDLY_OUT_0_a2_x_1_ (
        .combout(DDLY_OUT_0_a2_x_1),
        .dataa(PDL_WRITE_1),
```

```
        .datab(DDLY_OE),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam DDLY_OUT_0_a2_x_1_.operation_mode=”normal”;
defparam DDLY_OUT_0_a2_x_1_.output_mode=”comb_only”;
defparam DDLY_OUT_0_a2_x_1_.lut_mask=”8888”;
defparam DDLY_OUT_0_a2_x_1_.synch_mode=”off”;
defparam DDLY_OUT_0_a2_x_1_.sum_lutc_input=”datac”;
// @15:62
  cyclone_lcell DDLY_OUT_0_a2_x_2_ (
        .combout(DDLY_OUT_0_a2_x_2),
        .dataa(PDL_WRITE_2),
        .datab(DDLY_OE),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam DDLY_OUT_0_a2_x_2_.operation_mode=”normal”;
defparam DDLY_OUT_0_a2_x_2_.output_mode=”comb_only”;
defparam DDLY_OUT_0_a2_x_2_.lut_mask=”8888”;
defparam DDLY_OUT_0_a2_x_2_.synch_mode=”off”;
defparam DDLY_OUT_0_a2_x_2_.sum_lutc_input=”datac”;
// @15:62
  cyclone_lcell DDLY_OUT_0_a2_x_3_ (
        .combout(DDLY_OUT_0_a2_x_3),
        .dataa(PDL_WRITE_3),
        .datab(DDLY_OE),
        .datac(VCC),
        .datad(VCC),
```

```
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam DDLY_OUT_0_a2_x_3_.operation_mode=”normal”;
defparam DDLY_OUT_0_a2_x_3_.output_mode=”comb_only”;
defparam DDLY_OUT_0_a2_x_3_.lut_mask=”8888”;
defparam DDLY_OUT_0_a2_x_3_.synch_mode=”off”;
defparam DDLY_OUT_0_a2_x_3_.sum_lutc_input=”datac”;
// @15:62
  cyclone_lcell DDLY_OUT_0_a2_x_4_ (
        .combout(DDLY_OUT_0_a2_x_4),
        .dataa(PDL_WRITE_4),
        .datab(DDLY_OE),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam DDLY_OUT_0_a2_x_4_.operation_mode=”normal”;
defparam DDLY_OUT_0_a2_x_4_.output_mode=”comb_only”;
defparam DDLY_OUT_0_a2_x_4_.lut_mask=”8888”;
defparam DDLY_OUT_0_a2_x_4_.synch_mode=”off”;
defparam DDLY_OUT_0_a2_x_4_.sum_lutc_input=”datac”;
// @15:62
  cyclone_lcell DDLY_OUT_0_a2_x_5_ (
        .combout(DDLY_OUT_0_a2_x_5),
        .dataa(PDL_WRITE_5),
        .datab(DDLY_OE),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
```

```
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam DDLY_OUT_0_a2_x_5_.operation_mode=”normal”;
defparam DDLY_OUT_0_a2_x_5_.output_mode=”comb_only”;
defparam DDLY_OUT_0_a2_x_5_.lut_mask=”8888”;
defparam DDLY_OUT_0_a2_x_5_.synch_mode=”off”;
defparam DDLY_OUT_0_a2_x_5_.sum_lutc_input=”datac”;
// @15:62
  cyclone_lcell DDLY_OUT_0_a2_x_6_ (
        .combout(DDLY_OUT_0_a2_x_6),
        .dataa(PDL_WRITE_6),
        .datab(DDLY_OE),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam DDLY_OUT_0_a2_x_6_.operation_mode=”normal”;
defparam DDLY_OUT_0_a2_x_6_.output_mode=”comb_only”;
defparam DDLY_OUT_0_a2_x_6_.lut_mask=”8888”;
defparam DDLY_OUT_0_a2_x_6_.synch_mode=”off”;
defparam DDLY_OUT_0_a2_x_6_.sum_lutc_input=”datac”;
// @15:62
  cyclone_lcell DDLY_OUT_0_a2_x_7_ (
        .combout(DDLY_OUT_0_a2_x_7),
        .dataa(PDL_WRITE_7),
        .datab(DDLY_OE),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
```

```
        .regcascin(GND)
);
defparam DDLY_OUT_0_a2_x_7_.operation_mode=”normal”;
defparam DDLY_OUT_0_a2_x_7_.output_mode=”comb_only”;
defparam DDLY_OUT_0_a2_x_7_.lut_mask=”8888”;
defparam DDLY_OUT_0_a2_x_7_.synch_mode=”off”;
defparam DDLY_OUT_0_a2_x_7_.sum_lutc_input=”datac”;
endmodule /* pdl_if */

// VQM4.1+
module v1495usr_hal (
  DDLY_IN,
  DLO0_GATE,
  DLO1_GATE,
  FPGA_IN,
  GREEN_PULSE,
  LAD_IN,
  LCLK,
  PDL0_IN,
  PDL1_IN,
  PDL_DIR,
  PDL_SEL,
  PDL_WR,
  PDL_WRITE,
  PULSE,
  RED_PULSE,
  REG_DOUT,
  WnR,
  nADS,
  nBLAST,
  nLRESET,
  DDLY_OE,
  DDLY_OUT,
  DIRDDLY,
  DLO0_OUT,
  DLO1_OUT,
  FPGA_DIR,
  FPGA_OUT,
  LAD_OE,
  LAD_OUT,
  PDL0_OUT,
  PDL1_OUT,
  PDL_READ,
```

```
  REG_ADDR,
  REG_DIN,
  REG_RDEN,
  REG_WREN,
  START,
  USR_ACCESS,
  WR_DLY0,
  WR_DLY1,
  nINT,
  nLEDG,
  nLEDR,
  nOEDDLY0,
  nOEDDLY1,
  nREADY,
  nSTART
);
input [7:0] DDLY_IN ;
input DLO0_GATE ;
input DLO1_GATE ;
input [3:0] FPGA_IN ;
input GREEN_PULSE ;
input [15:0] LAD_IN ;
input LCLK ;
input PDL0_IN ;
input PDL1_IN ;
input PDL_DIR ;
input PDL_SEL ;
input PDL_WR ;
input [7:0] PDL_WRITE ;
input [3:0] PULSE ;
input RED_PULSE ;
input [15:0] REG_DOUT ;
input WnR ;
input nADS ;
input nBLAST ;
input nLRESET ;
output DDLY_OE ;
output [7:0] DDLY_OUT ;
output DIRDDLY ;
output DLO0_OUT ;
output DLO1_OUT ;
output [3:0] FPGA_DIR ;
output [3:0] FPGA_OUT ;
```

```
output LAD_OE ;
output [15:0] LAD_OUT ;
output PDL0_OUT ;
output PDL1_OUT ;
output [7:0] PDL_READ ;
output [15:0] REG_ADDR ;
output [15:0] REG_DIN ;
output REG_RDEN ;
output REG_WREN ;
output [1:0] START ;
output USR_ACCESS ;
output WR_DLY0 ;
output WR_DLY1 ;
output nINT ;
output nLEDG ;
output nLEDR ;
output nOEDDLY0 ;
output nOEDDLY1 ;
output nREADY ;
output [3:2] nSTART ;
wire DLO0_GATE ;
wire DLO1_GATE ;
wire GREEN_PULSE ;
wire LCLK ;
wire PDL0_IN ;
wire PDL1_IN ;
wire PDL_DIR ;
wire PDL_SEL ;
wire PDL_WR ;
wire RED_PULSE ;
wire WnR ;
wire nADS ;
wire nBLAST ;
wire nLRESET ;
wire DDLY_OE ;
wire DIRDDLY ;
wire DLO0_OUT ;
wire DLO1_OUT ;
wire LAD_OE ;
wire PDL0_OUT ;
wire PDL1_OUT ;
wire REG_RDEN ;
wire REG_WREN ;
```

```
wire USR_ACCESS ;
wire WR_DLY0 ;
wire WR_DLY1 ;
wire nINT ;
wire nLEDG ;
wire nLEDR ;
wire nOEDDLY0 ;
wire nOEDDLY1 ;
wire nREADY ;
wire [2:0] I8_TCNT1;
wire [7:0] I8_TCNT2;
wire GND ;
wire VCC ;
wire nLRESET_i_x_i ;
wire sclrsclrI8_un7_tcnt1_a ;
wire sclrsclrI8_un14_tcnt3_5_5 ;
wire sclrsclrI8_un14_tcnt3_5_4 ;
//@1:1
  assign VCC = 1'b1;
//@1:1
  assign GND = 1'b0;
  assign  nLRESET_i_x_i = ~ nLRESET;
  cyclone_lcell DLO0_GATE_i_x (
        .combout(nSTART[2]),
        .dataa(DLO0_GATE),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam DLO0_GATE_i_x.operation_mode="normal";
defparam DLO0_GATE_i_x.output_mode="comb_only";
defparam DLO0_GATE_i_x.lut_mask="5555";
defparam DLO0_GATE_i_x.synch_mode="off";
defparam DLO0_GATE_i_x.sum_lutc_input="datac";
  cyclone_lcell DLO1_GATE_i_x (
        .combout(nSTART[3]),
```

```
        .dataa(DLO1_GATE),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam DLO1_GATE_i_x.operation_mode=”normal”;
defparam DLO1_GATE_i_x.output_mode=”comb_only”;
defparam DLO1_GATE_i_x.lut_mask=”5555”;
defparam DLO1_GATE_i_x.synch_mode=”off”;
defparam DLO1_GATE_i_x.sum_lutc_input=”datac”;
  cyclone_lcell PDL_SEL_i_x (
        .combout(nOEDDLY1),
        .dataa(nOEDDLY0),
        .datab(VCC),
        .datac(VCC),
        .datad(VCC),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam PDL_SEL_i_x.operation_mode=”normal”;
defparam PDL_SEL_i_x.output_mode=”comb_only”;
defparam PDL_SEL_i_x.lut_mask=”5555”;
defparam PDL_SEL_i_x.synch_mode=”off”;
defparam PDL_SEL_i_x.sum_lutc_input=”datac”;
// @13:117
  cyclone_lcell sclrsclrI8_un7_tcnt1_a_cZ (
        .combout(sclrsclrI8_un7_tcnt1_a),
        .dataa(I8_TCNT1[0]),
        .datab(I8_TCNT1[1]),
        .datac(I8_TCNT1[2]),
        .datad(VCC),
```

```
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam sclrsclrI8_un7_tcnt1_a_cZ.operation_mode="normal";
defparam sclrsclrI8_un7_tcnt1_a_cZ.output_mode="comb_only";
defparam sclrsclrI8_un7_tcnt1_a_cZ.lut_mask="0101";
defparam sclrsclrI8_un7_tcnt1_a_cZ.synch_mode="off";
defparam sclrsclrI8_un7_tcnt1_a_cZ.sum_lutc_input="datac";
// @13:129
  cyclone_lcell sclrsclrI8_un14_tcnt3_5_5_cZ (
        .combout(sclrsclrI8_un14_tcnt3_5_5),
        .dataa(I8_TCNT2[6]),
        .datab(I8_TCNT2[7]),
        .datac(I8_TCNT2[4]),
        .datad(I8_TCNT2[5]),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam sclrsclrI8_un14_tcnt3_5_5_cZ.operation_mode="normal";
defparam sclrsclrI8_un14_tcnt3_5_5_cZ.output_mode="comb_only";
defparam sclrsclrI8_un14_tcnt3_5_5_cZ.lut_mask="0001";
defparam sclrsclrI8_un14_tcnt3_5_5_cZ.synch_mode="off";
defparam sclrsclrI8_un14_tcnt3_5_5_cZ.sum_lutc_input="datac";
// @13:129
  cyclone_lcell sclrsclrI8_un14_tcnt3_5_4_cZ (
        .combout(sclrsclrI8_un14_tcnt3_5_4),
        .dataa(I8_TCNT2[2]),
        .datab(I8_TCNT2[3]),
        .datac(I8_TCNT2[0]),
        .datad(I8_TCNT2[1]),
        .aclr(GND),
        .sclr(GND),
        .sload(GND),
```

```
        .ena(VCC),
        .inverta(GND),
        .aload(GND),
        .regcascin(GND)
);
defparam sclrsclrI8_un14_tcnt3_5_4_cZ.operation_mode=”normal”;
defparam sclrsclrI8_un14_tcnt3_5_4_cZ.output_mode=”comb_only”;
defparam sclrsclrI8_un14_tcnt3_5_4_cZ.lut_mask=”0001”;
defparam sclrsclrI8_un14_tcnt3_5_4_cZ.synch_mode=”off”;
defparam sclrsclrI8_un14_tcnt3_5_4_cZ.sum_lutc_input=”datac”;
// @17:340
  led_if I8 (
        .TCNT2_7(I8_TCNT2[7]),
        .TCNT2_6(I8_TCNT2[6]),
        .TCNT2_5(I8_TCNT2[5]),
        .TCNT2_4(I8_TCNT2[4]),
        .TCNT2_3(I8_TCNT2[3]),
        .TCNT2_2(I8_TCNT2[2]),
        .TCNT2_1(I8_TCNT2[1]),
        .TCNT2_0(I8_TCNT2[0]),
        .TCNT1_2(I8_TCNT1[2]),
        .TCNT1_1(I8_TCNT1[1]),
        .TCNT1_0(I8_TCNT1[0]),
        .nLRESET_i_x_i(nLRESET_i_x_i),
        .nLEDG(nLEDG),
        .nLEDR(nLEDR),
        .GREEN_PULSE(GREEN_PULSE),
        .RED_PULSE(RED_PULSE),
        .nLRESET(nLRESET),
        .un7_tcnt1_a(sclrsclrI8_un7_tcnt1_a),
        .un14_tcnt3_5_5(sclrsclrI8_un14_tcnt3_5_5),
        .un14_tcnt3_5_4(sclrsclrI8_un14_tcnt3_5_4),
        .LCLK(LCLK)
);
// @17:349
  localbusif I1 (
        .REG_DOUT_15(REG_DOUT[15]),
        .REG_DOUT_14(REG_DOUT[14]),
        .REG_DOUT_13(REG_DOUT[13]),
        .REG_DOUT_12(REG_DOUT[12]),
        .REG_DOUT_11(REG_DOUT[11]),
        .REG_DOUT_10(REG_DOUT[10]),
        .REG_DOUT_9(REG_DOUT[9]),
```

```
.REG_DOUT_8(REG_DOUT[8]),
.REG_DOUT_7(REG_DOUT[7]),
.REG_DOUT_6(REG_DOUT[6]),
.REG_DOUT_5(REG_DOUT[5]),
.REG_DOUT_4(REG_DOUT[4]),
.REG_DOUT_3(REG_DOUT[3]),
.REG_DOUT_2(REG_DOUT[2]),
.REG_DOUT_1(REG_DOUT[1]),
.REG_DOUT_0(REG_DOUT[0]),
.LAD_OUT_0_a4_x_15(LAD_OUT[15]),
.LAD_OUT_0_a4_x_14(LAD_OUT[14]),
.LAD_OUT_0_a4_x_13(LAD_OUT[13]),
.LAD_OUT_0_a4_x_12(LAD_OUT[12]),
.LAD_OUT_0_a4_x_11(LAD_OUT[11]),
.LAD_OUT_0_a4_x_10(LAD_OUT[10]),
.LAD_OUT_0_a4_x_9(LAD_OUT[9]),
.LAD_OUT_0_a4_x_8(LAD_OUT[8]),
.LAD_OUT_0_a4_x_7(LAD_OUT[7]),
.LAD_OUT_0_a4_x_6(LAD_OUT[6]),
.LAD_OUT_0_a4_x_5(LAD_OUT[5]),
.LAD_OUT_0_a4_x_4(LAD_OUT[4]),
.LAD_OUT_0_a4_x_3(LAD_OUT[3]),
.LAD_OUT_0_a4_x_2(LAD_OUT[2]),
.LAD_OUT_0_a4_x_1(LAD_OUT[1]),
.LAD_OUT_0_a4_x_0(LAD_OUT[0]),
.LAD_IN_15(REG_DIN[15]),
.LAD_IN_14(REG_DIN[14]),
.LAD_IN_13(REG_DIN[13]),
.LAD_IN_12(REG_DIN[12]),
.LAD_IN_11(REG_DIN[11]),
.LAD_IN_10(REG_DIN[10]),
.LAD_IN_9(REG_DIN[9]),
.LAD_IN_8(REG_DIN[8]),
.LAD_IN_7(REG_DIN[7]),
.LAD_IN_6(REG_DIN[6]),
.LAD_IN_5(REG_DIN[5]),
.LAD_IN_4(REG_DIN[4]),
.LAD_IN_3(REG_DIN[3]),
.LAD_IN_2(REG_DIN[2]),
.LAD_IN_1(REG_DIN[1]),
.LAD_IN_0(REG_DIN[0]),
.REG_ADDR_i_m2_x_15(REG_ADDR[15]),
.REG_ADDR_i_m2_x_14(REG_ADDR[14]),
```

```
        .REG_ADDR_i_m2_x_13(REG_ADDR[13]),
        .REG_ADDR_i_m2_x_12(REG_ADDR[12]),
        .REG_ADDR_i_m2_x_11(REG_ADDR[11]),
        .REG_ADDR_i_m2_x_10(REG_ADDR[10]),
        .REG_ADDR_i_m2_x_9(REG_ADDR[9]),
        .REG_ADDR_i_m2_x_8(REG_ADDR[8]),
        .REG_ADDR_i_m2_x_7(REG_ADDR[7]),
        .REG_ADDR_i_m2_x_6(REG_ADDR[6]),
        .REG_ADDR_i_m2_x_5(REG_ADDR[5]),
        .REG_ADDR_i_m2_x_4(REG_ADDR[4]),
        .REG_ADDR_i_m2_x_3(REG_ADDR[3]),
        .REG_ADDR_i_m2_x_2(REG_ADDR[2]),
        .REG_ADDR_i_m2_x_1(REG_ADDR[1]),
        .REG_ADDR_i_m2_x_0(REG_ADDR[0]),
        .REG_RDEN_i_x(REG_RDEN),
        .nREADY_s_i_i_x(nREADY),
        .un2_lad_oe_i_0_a2_x(LAD_OE),
        .WnR(WnR),
        .WREN_s(REG_WREN),
        .nBLAST(nBLAST),
        .nLRESET(nLRESET),
        .nADS(nADS),
        .LCLK(LCLK)
);
// @17:368
  pdl_if I5 (
        .PDL_WRITE_7(PDL_WRITE[7]),
        .PDL_WRITE_6(PDL_WRITE[6]),
        .PDL_WRITE_5(PDL_WRITE[5]),
        .PDL_WRITE_4(PDL_WRITE[4]),
        .PDL_WRITE_3(PDL_WRITE[3]),
        .PDL_WRITE_2(PDL_WRITE[2]),
        .PDL_WRITE_1(PDL_WRITE[1]),
        .PDL_WRITE_0(PDL_WRITE[0]),
        .DDLY_OUT_0_a2_x_7(DDLY_OUT[7]),
        .DDLY_OUT_0_a2_x_6(DDLY_OUT[6]),
        .DDLY_OUT_0_a2_x_5(DDLY_OUT[5]),
        .DDLY_OUT_0_a2_x_4(DDLY_OUT[4]),
        .DDLY_OUT_0_a2_x_3(DDLY_OUT[3]),
        .DDLY_OUT_0_a2_x_2(DDLY_OUT[2]),
        .DDLY_OUT_0_a2_x_1(DDLY_OUT[1]),
        .DDLY_OUT_0_a2_x_0(DDLY_OUT[0]),
        .DDLY_OE(DIRDDLY)
```

```
);
assign PDL_READ[0] = DDLY_IN[0];
assign PDL_READ[1] = DDLY_IN[1];
assign PDL_READ[2] = DDLY_IN[2];
assign PDL_READ[3] = DDLY_IN[3];
assign PDL_READ[4] = DDLY_IN[4];
assign PDL_READ[5] = DDLY_IN[5];
assign PDL_READ[6] = DDLY_IN[6];
assign PDL_READ[7] = DDLY_IN[7];
assign REG_DIN[0] = LAD_IN[0];
assign REG_DIN[1] = LAD_IN[1];
assign REG_DIN[2] = LAD_IN[2];
assign REG_DIN[3] = LAD_IN[3];
assign REG_DIN[4] = LAD_IN[4];
assign REG_DIN[5] = LAD_IN[5];
assign REG_DIN[6] = LAD_IN[6];
assign REG_DIN[7] = LAD_IN[7];
assign REG_DIN[8] = LAD_IN[8];
assign REG_DIN[9] = LAD_IN[9];
assign REG_DIN[10] = LAD_IN[10];
assign REG_DIN[11] = LAD_IN[11];
assign REG_DIN[12] = LAD_IN[12];
assign REG_DIN[13] = LAD_IN[13];
assign REG_DIN[14] = LAD_IN[14];
assign REG_DIN[15] = LAD_IN[15];
assign START[0] = PDL0_IN;
assign START[1] = PDL1_IN;
assign DIRDDLY = PDL_DIR;
assign nOEDDLY0 = PDL_SEL;
assign WR_DLY1 = PDL_WR;
assign PDL0_OUT = PULSE[0];
assign PDL1_OUT = PULSE[1];
assign DLO0_OUT = PULSE[2];
assign DLO1_OUT = PULSE[3];
assign DDLY_OE = DIRDDLY;
assign FPGA_DIR[0] = GND;
assign FPGA_DIR[1] = GND;
assign FPGA_DIR[2] = GND;
assign FPGA_DIR[3] = GND;
assign FPGA_OUT[0] = GND;
assign FPGA_OUT[1] = GND;
assign FPGA_OUT[2] = GND;
assign FPGA_OUT[3] = GND;
```

```
assign USR_ACCESS = VCC;
assign WR_DLY0 = WR_DLY1;
assign nINT = VCC;
endmodule /* v1495usr_hal */
```